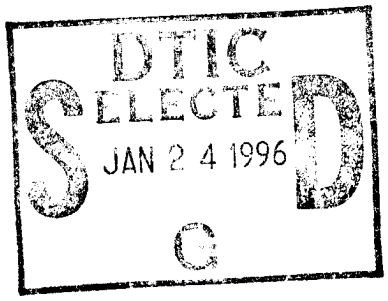
RL-TR-95-143
Final Technical Report
August 1995

GENERIC TOOLS FOR TRANSPORTATION PLANNING AND SCHEDULING

Kestrel Institute



Sponsored by
Advanced Research Projects Agency
ARPA Order No. 7730

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-143 has been reviewed and is approved for publication.

APPROVED:

JOHN F. LEMMER
Project Engineer

FOR THE COMMANDER:

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (C3CA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

GENERIC TOOLS FOR TRANSPORTATION PLANNING AND SCHEDULING

Douglas R. Smith
Eduardo Parra
Stephen J. Westfold

Contractor: Kestrel Institute
Contract Number: F30602-91-C-0043
Effective Date of Contract: 07 Mar 1991
Contract Expiration Date: 30 September 1994
Short Title of Work: Generic Tools for Transportation Planning and Scheduling
Period of Work Covered: Mar 91 - Sep 94

Principal Investigator: Douglas R. Smith
Phone: (415) 493-6871

RL Project Engineer: John F. Lemmer
Phone: (315) 330-3655

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by John F. Lemmer, RK (C3CA), 525 Brooks Rd, Griffiss AFB NY 13441-4505.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1995	Final Mar 91 - Sep 94

4. TITLE AND SUBTITLE
GENERIC TOOLS FOR TRANSPORTATION PLANNING AND SCHEDULING

5. FUNDING NUMBERS
C - F30602-91-C-0043
PE - 62301E
PR - G730
TA - 00
WU - 11

6. AUTHOR(S)
Douglas R. Smith, Eduardo A. Parra, and Stephen J. Westfold

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Kestrel Institute
3260 Hillview Ave
Palo Alto CA 94304

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

Rome Laboratory (OCPC)
25 Electronic Pky
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-95-143

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: John F. Lemmer/C3CA/(315) 330-3655

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes our research on transportation planning and scheduling supported by the ARPA/Rome Lab Planning Initiative (ARPI). The main goal of this project was to develop generic tools to support the construction of flexible, high-performance planning and scheduling software. Our technical approach is based on program transformation technology that allow the systematic machine-supported development of software from requirement specifications. The development process can produce highly efficient code along with a proof of the code's correctness.

We have used KIDS (Kestrel Interactive Development System) to derive extremely fast and accurate transportation schedulers from formal specifications. As test data, we use strategic transportation plans that are generated by US government planners. A typical problem with 10,000 movement requirements takes the derived scheduler 1-3 minutes to solve compared with 2.5 hours for a deployed feasibility estimator (JFAST) and 36 hours for deployed schedulers (FLOGEN, ADANS). The computed schedules use relatively few resources and satisfy all specified constraints. The speed of this scheduler is due to the synthesis of strong constraint checking and constraint propagation code.

14. SUBJECT TERMS
Strategic transportation planning, Transportation scheduling, Formal specification, Planning, Scheduling software

15. NUMBER OF PAGES
114

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Contents

List of Figures

1 Executive Summary

This report describes our research on transportation planning and scheduling supported by the ARPA/Rome Lab Planning Initiative (ARPI). The main goal of this project was to develop generic tools to support the construction of flexible, high-performance planning and scheduling software. Our technical approach is based on *program transformation* technology which allows the systematic machine-supported development of software from requirement specifications. The development process can produce highly efficient code along with a proof of the code's correctness.

Our approach to developing scheduling software involves several stages. The first step is to develop a formal model of the transportation scheduling domain, called a *domain theory.* Second, the constraints, objectives, and preferences of a particular scheduling problem are stated within a domain theory as a *problem specification.* Finally, an executable scheduler is produced semi-automatically by applying a sequence of *transformations* to the problem specification. The transformations embody programming knowledge about algorithms, data structures, program optimization techniques, etc. The result of the transformation process is executable code that is consistent with the problem specification.

The U.S. Transportation Command and the component service commands use a relational database scheme called a TPFDD (Time-Phased Force and Deployment Data) for specifying the transportation requirements of an operation, such as Desert Storm or the Somalia relief effort. We developed a domain theory of TPFDD scheduling defining the concepts of this problem and developed laws for reasoning about them. A program transformation system called KIDS (Kestrel Interactive Development System) was used to derive and optimize a variety of global search scheduling algorithms that perform constraint propagation [37]. The resulting code, generically called KTS (Kestrel Transportation Scheduler), has been run on a variety of TPFDDs generated by planners at USTRANSCOM and other sites. With one such TPFDD problem, KTS was able to schedule over 15,000 individual movement requirements in about 5 minutes. The schedule used relatively few resources and satisfied all specified constraints. KTS is orders of magnitude faster than any other TPFDD scheduler known to us.

In 1994 we began to develop a scheduler to support PACAF (Pacific Air Force) at Hickham AFB, Honolulu which is tasked with in-theater scheduling of a fleet of 26 C-130 cargo aircraft in the Pacific region. Several variants of a theater scheduler (called ITAS for In-Theater Airlift Scheduler) have been installed at PACAF, and more are planned. ITAS runs on an Apple Powerbook laptop computer which makes it attractive both for field and command center operations. ITAS can currently produce ATOs (Air Tasking Orders) based on the schedules that it generates. The most recent version simultaneously schedules the following classes of resources: (1) aircraft, (2) aircrews and their duty day cycles, (3) ground crews for unloading, and (4) ramp space at ports.

2 Introduction

This report describes our research on the transformational development of transportation plans and schedules. Our approach to developing scheduling software involves several stages. The first step is to develop a formal model of the transportation scheduling domain, called a *domain theory*. Second, the constraints, objectives, and preferences of a particular scheduling problem are stated within a domain theory as a *problem specification*. Finally, an executable scheduler is produced semi-automatically by applying a sequence of *transformations* to the problem specification. The transformations embody programming knowledge about algorithms, data structures, program optimization techniques, etc. The result of the transformation process is executable code that is consistent with the given problem specification. Furthermore, the resulting code can be extremely efficient.

Transportation scheduling tools currently used by the U.S. government are based on models of the transportation domain that few people understand [11]. Consequently, users often do not trust that the scheduling results reflect the characteristics of the current situation. Our approach tries to address this issue by making the domain model and scheduling problem explicit and clear. If a scheduling situation arises which is not treated by existing scheduling tools, the user can specify the problem and generate a situation-specific scheduler.

One of the benefits of a transformational approach to scheduling is the synthesis of specialized constraint management code. Previous systems for performing scheduling in AI (e.g. [13, 12, 48, 47]) and Operations Research [2, 24] use constraint representations and operations that are geared for a broad class of problems, such as constraint satisfaction problems or linear programs. In contrast, transformational techniques can derive specialized representations for constraints and related data, and also derive efficient specialized code for constraint checking and constraint propagation.

Figure 1 describes our vision of an advanced environment for producing planning/scheduling software. Briefly, the idea is to rapidly develop a situation-specific domain model and problem specification using a knowledge-elicitation system, and then to *synthesize* high-performance planning and scheduling tools that are specialized to the current situation. The majority of users' interaction would be codifying the *domain theory* and specification of the current situation, to aid in synthesizing a customized planning/scheduling tool.

We now step through the process in more detail. Several classes of users are involved in the construction and use of a scheduling system.

One class of users, who include domain experts and specialists in model construction, interact with a knowledge elicitation system to help classify the features of the situation and select, compose, extend, and refine, (possibly abstract) models from a preexisting library of domain models. The result is a model and problem specification tailored to the details of the current situation (as closely as expertise and time permit).

Another class of users, who specialize in software design and formal modeling of programming knowledge, interact with a planning/scheduling synthesis system to develop code from the problem specification. The interaction involves composing components from a library of reusable parts, or selecting and applying representations of abstract programming knowledge

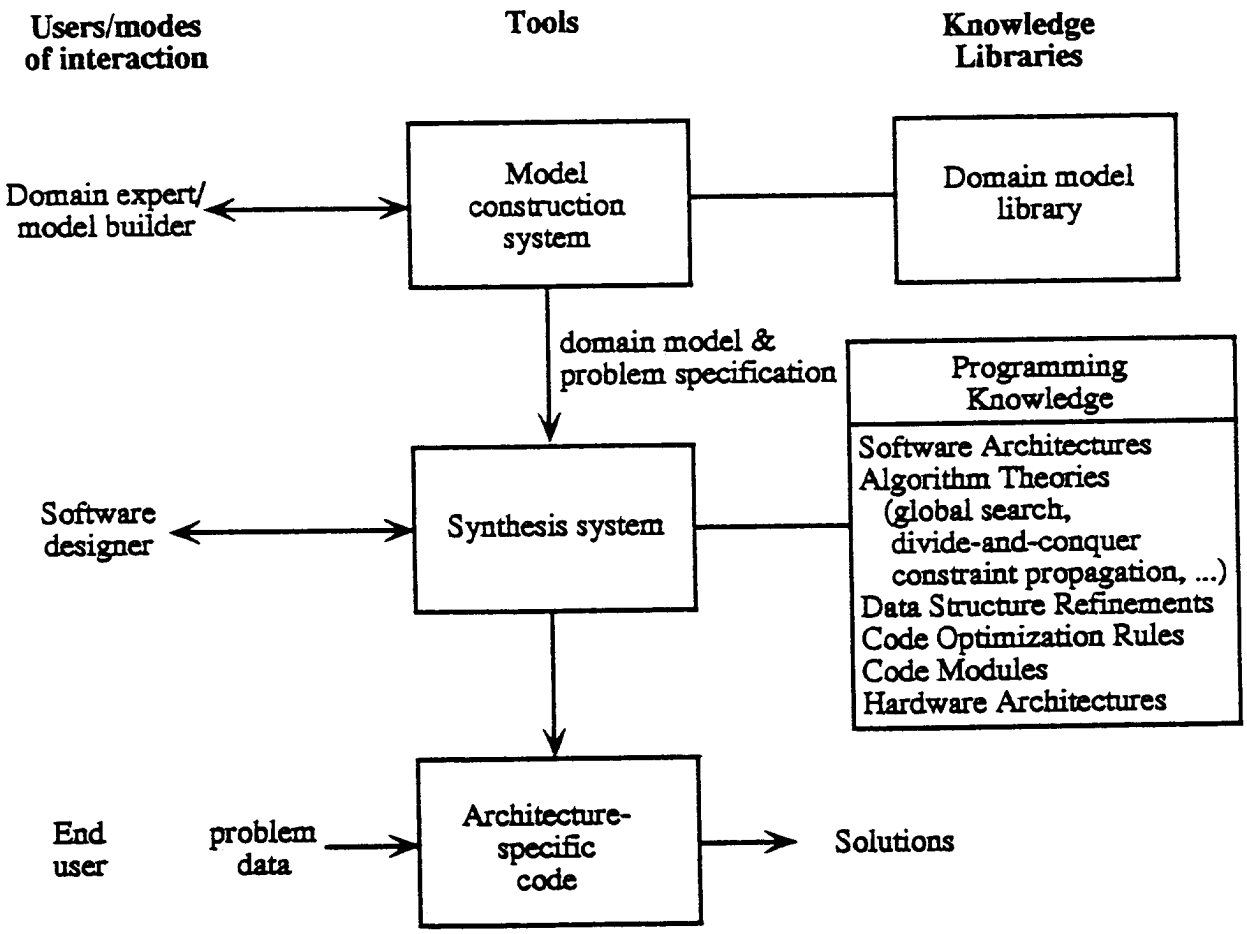


Figure 1: Advanced Development Environment for Planning and Scheduling Software

about algorithms, data structures, code optimization techniques, planning and scheduling-specific design strategies, and so on. Most of the design process is automated, with only a few high-level design decisions made by the developer. Another interactive task for this user is the evolution of the programming knowledge-base itself.

The output of the synthesis system is executable planning/scheduling code which can then be embedded in a planning/scheduling system and executed by an end-user.

There is a feedback loop implicit in the figure. The end user/domain expert using the synthesized code may detect missing constraints, or have preferences or other information not accounted for in the code. This information is fed back to the model-building stage and the process iterates. The fact that each synthesis step preserves consistency between problem specification and generated code means that *maintenance and evolution back up to the specification/domain model level*, not the code-level as in current practice.

We developed an approximation to this vision in the current project, based on the KIDS system, and demonstrated its feasibility.

In Sections 3 through 5 we present KIDS and the process of developing domain theories, specifications, and code for scheduling problems. Section 7 describes fundamental work underlying our technical approach. The Appendix describes work performed under subcontract to ADS that focused on issues in planning and resource allocation.

3 Summary of Results

- *TPFDD Scheduling* – The U.S. Transportation Command and the component service commands use a relational database scheme called a TPFDD (Time-Phased Force and Deployment Data) for specifying the transportation requirements of an operation, such as Desert Storm or the Somalia relief effort. We developed a domain theory of TPFDD scheduling defining the concepts of this problem and developed laws for reasoning about them. KIDS (Kestrel Interactive Development System) was used to derive and optimize a variety of global search scheduling algorithms that perform constraint propagation [37]. The resulting code, generically called KTS (Kestrel Transportation Scheduler), has been run on a variety of TPFDDs generated by planners at USTRANSCOM and other sites.

  We compared the performance of KTS with several other TPFDD scheduling systems: JFAST, FLOGEN, DITOPS, and PFE. We do not have access to JFAST and FLOGEN, but these are (or were) operational tools at AMC (Airlift Mobility Command, Scott AFB). According to [11] and David Brown (retired military planner consulting with the Planning Initiative), on a typical TPFDD of about 10,000 movement records, JFAST takes several hours and FLOGEN about 36 hours. KTS on a TPFDD of this size will produce a detailed schedule in *one to three minutes*. So KTS seems to be a factor of about 25 times faster than JFAST and over 250 times faster than FLOGEN. The currently operational ADANS system reportedly runs at about the same speed as FLOGEN. KTS is orders of magnitude faster than any other TPFDD scheduler known to us.

Comparison with PFE: On the MEDCOM-SITUATION from the CPE, KTS is about 5 times faster than PFE and produces a SEA schedule with only 14% of the delay of the PFE schedule. KTS also produces a far more accurate estimate of the planes needed to handle the AIR movements, since PFE is only estimating feasibility whereas KTS produces a detailed schedule.

- *KTS system*

  The KTS code is embedded in a CLIM interface adapted from the PFE interface built by BBN. It allows selecting a variety of different TPFDDs to schedule, real-time editing of resource models and situation models, dynamic rescheduling, and graphical tools for browsing and analyzing the resulting schedules (e.g. Unit and Resource Gantt charts, closure graphs, statistics, etc.). This code has been available via ftp to ARPI projects since 1993.

- *Contributions to the Planning Initiative*

  We integrated a standalone version of KTS into the CPE. KTS is an alternate deployment "simulator" to PFE and takes the same input data (geographical information, deployment plan, and situation). The results of the scheduling process (feasibility analysis) are written back to the KS for use by the planning components in case replanning is needed. Currently, this analysis object contains the same kind information as PFE (e.g. FAD, SLD, etc.). Results may also be displayed on the screen using charts similar to those available in the CDART system. Comparison between KTS and PFE is detailed in Section 6.3.

  Other PI-related contributions include participation on the SWAT-B team and other committees, contributions to the IEEE Expert special issue on the Planning Initiative, presentations at workshops, and site visits to USTC, AMC, McGuire AFB, and PACAF (Hickham AFB).

- *Theater Scheduling*

  In 1994 we began to develop a scheduler to support PACAF (Pacific Air Force) at Hickham AFB, Honolulu which is tasked with in-theater scheduling of a fleet of 26 C-130 cargo aircraft in the Pacific region. We developed (and are continuing to evolve) a theory of theater transportation scheduling. Several variants of a theater scheduler (called ITAS for In-Theater Airlift Scheduler) have been developed to date, and more are planned. The interface to ITAS and integration with a commercial database package have been developed by BBN. ITAS runs on an Apple Powerbook laptop computer. The laptop platform makes it attractive both for field and command center operations. ITAS can currently produce ATOs (Air Tasking Orders) based on the schedules that it generates.

  The ITAS schedulers have emphasized flexibility and rich constraint modeling. Versions of ITAS were installed at PACAF in August 1994, September 1994, and February 1995. The most recent version simultaneously schedules the following classes of resources: (1) aircraft, (2) aircrews and their duty day cycles, (3) ground crews for unloading, and (4) ramp space at ports.

  One of the reasons for the interest of PACAF in this project, is to capture some of the knowledge and experience of skilled personnel before they retire or are rotated.

- *Synthesis of constraint propagation code*

  A key technical achievement of this project was discovering and implementing technology for generating efficient constraint propagation code. The speed of the KTS schedulers derives from the extremely fast checking and propagation of constraint information at every node of the runtime search tree. Whereas some knowledge-based approaches to scheduling will search a tree at the rate of several nodes per second, our synthesized schedulers search *several hundred thousand* nodes per second.

  Briefly, the idea is to derive necessary conditions on feasibility of a candidate schedule. These conditions are called *cutting constraints*. The derived cutting constraints for a particular scheduling problem are analyzed to produce code that iteratively fixes violated constraints until the cutting constraints are satisfied. This iterative process subsumes the well-known processes of constraint propagation in the AI literature and the notion of cutting planes from the Operations Research literature [40, 46]. Constraint propagation is discussed in more detail in Section 6.2.3.

- *Classification approach to design*

  We developed a new approach to the problem of how to construct refinements of specifications formally and incrementally. The idea is to use a taxonomy of abstract design concepts, each represented by a *design theory*. An abstract design concept is applied by constructing a specification morphism from its design theory to a requirement specification. Procedures for computing colimits and for constructing specification morphisms provide computational support for this approach. Although the classification approach applies to the incremental application of *any* kind of knowledge formally represented in a hierarchy of theories, our work mainly focused on a hierarchy of algorithm design theories and its applications to logistical applications [39, 38]. This technique enable us to integrate at a deep semantic level problem-solving methods from Computer Science (e.g. divide-and-conquer, global search), Artificial Intelligence (e.g. heuristic search, constraint propagation, neural nets), and Operations Research (e.g. Simplex, integer programming, network algorithms). Classification is discussed in more detail in Section 7.

4 KIDS model of program development

KIDS is a program transformation system – one applies a sequence of consistency-preserving transformations to an initial specification and achieves a correct and hopefully efficient program [42]. The system emphasizes the application of complex high-level transformations that perform significant and meaningful actions. From the user's point of view the system allows the user to make high-level design decisions like, "design a divide-and-conquer algorithm for that specification" or "simplify that expression in context". We hope that decisions at this level will be both intuitive to the user and be high-level enough that useful programs can be derived within a reasonable number of steps.

The user typically goes through the following steps in using KIDS for program development.

1. *Develop a domain theory* – An application domain is modeled by a domain theory (a collection of types, operations, laws, and inference rules). The domain theory specifies the concepts, operations, and relationships that characterize the application and supports reasoning about the domain via a deductive inference system. Our experience has been that distributive and monotonicity laws provide most of the laws that are needed to support design and optimization of code. KIDS has a theory development component that supports the automated derivation of various kinds of laws.

2. *Create a specification* – The user enters a problem specification stated in terms of the underlying domain theory.

3. *Apply a design tactic* – The user selects an algorithm design tactic from a menu and applies it to a specification. Currently KIDS has tactics for simple problem reduction (reducing a specification to a library routine) [33], divide-and-conquer [33], global search (binary search, backtrack, branch-and-bound) [34], problem reduction generators (dynamic programming, general branch-and-bound, and game-tree search algorithms) [36], and local search (hillclimbing algorithms) [23].

4. *Apply optimizations* – The KIDS system allows the application of optimization techniques such as expression simplification, partial evaluation, finite differencing, case analysis, and other transformations [42]. The user selects an optimization method from a menu and applies it by pointing at a program expression. Each of the optimization methods are fully automatic and, with the exception of simplification (which is arbitrarily hard), take only a few seconds.

5. *Apply data type refinements* – The user can select implementations for the high-level data types in the program. Data type refinement rules carry out the details of constructing the implementation [5].

6. *Compile* – The resulting code is compiled to executable form. In a sense, KIDS can be regarded as a front-end to a conventional compiler.

Actually, the user is free to apply any subset of the KIDS operations in any order – the above sequence is typical of our experiments in algorithm design. A new system, called Specware, is currently under construction at Kestrel as a successor to KIDS. Specware is based on concepts of higher-order algebraic specifications, morphisms, and categorical constructions [20, 38, 39, 50].

5 Specifying a Scheduler

5.1 What is Scheduling?

The essential notion of scheduling is that certain activities are assigned to resources over certain time intervals. Various constraints on the assignments must be satisfied and certain measures of the cost or "goodness" of the assignment are to be optimized.

A domain theory for scheduling defines the basic concepts of scheduling and the laws for reasoning about the concepts. After a review of the relevant literature (e.g. [12]) we have identified the following general components of a scheduling domain theory.

1. *Activities* – A model of the activities can include their internal structure and characteristics, hierarchies of activity abstractions, and various operations on activities.

2. *Resources* – A model of the resources can include their internal structure and characteristics, hierarchies of resource abstractions, and various operations on resources.

3. *Time* – A time model can include a calculus of time-points or time-intervals [1, 21].

4. *Constraints* – A constraint model includes the language for stating constraints and a calculus for reasoning about them. Several classes of constraints commonly arise in practice. The most common are *precedence constraints* (which state that one activity must precede another) and *capacity constraints* (which state bounds on the capacities of resources). A constraint calculus is used to analyze constraints and to propagate the effects of new constraints through a given constraint set. Fox et al. also identify physical constraints, organizational constraints, preferences, enablement constraints, and availability constraints.

5. *Objectives* – Typically we seek to minimize the cost of a schedule. Cost can be measured in terms of time to completion, work-in-progress, total cost of consumed resources, and so on.

6. *Scheduling problem* – Using the above concepts we can formulate a variety of scheduling problems. A reservation is a triple consisting of an activity, a resource, and a time interval. Generally, a schedule is a set of *reservations* that satisfy a collection of constraints and optimize (or produce a reasonably good value of) the objective.

$$\{ \langle \textit{activity}, \textit{resource}, \textit{time interval} \rangle \mid \textit{constraints} \}.$$

Many scheduling problem intancess are overconstrained – there are too few resources to schedule the activities and satisfy all constraints. Usually overconstrained problems are dealt with by relaxing the constraints and trying to satisfy as many of the constraints as possible. The usual method is to move constraints into the objective function. This entails reformulating the constraint so that it yields a quantitative measure of how well it has been satisfied. See further discussion in Section 6.2.4.

5.2 Strategic Transportation Scheduling

Transportation scheduling specializes the above general notion of scheduling: activities correspond to *movement requirements* and resources correspond to transportation assets such as planes, ships, and trucks.

A typical movement requirement has the following information:

move–type : *movement–type*	$\mapsto$	*BULK–MOVEMENT*
quantity : *integer*	$\mapsto$	2 (STONS - Short TONS)
release–date : *time*	$\mapsto$	0 (seconds from C-date)
due–date : *time*	$\mapsto$	86400 (seconds from C-date)
poe : *port*	$\mapsto$	*UHHZ*
pod : *port*	$\mapsto$	*VRJT*
distance : *integer*	$\mapsto$	5340 (nautical miles)
mode : *symbol*	$\mapsto$	*AIR*

Here quantity for AIR movements is in short tons (STONs); the release and due dates are in seconds starting from C-DATE; poe (port of embarkation) and pod (port of debarkation) are given by code names; distance is in nautical miles, and the transportation mode is either AIR or SEA. A collection of movement requirements is called a TPFDD (Time-Phased Force Deployment Data).

Resources are characterized by their capacities (both passenger (PAX) and cargo capacities), and travel rate in knots.

As an example, we used a small dataset extracted from a TUNISIA TPFDD created at AFSC. This problem instance involves 480 movement requirements from 20 airports and 3 seaports to 8 airports and 2 seaports. Available air resources include KC10s, C-141s, C-5s and sea resources include tankers (small, medium, and large), RO-ROs, LASHs, sea barges, containerships, and breakbulks.

Eleven constraints characterize a feasible schedule for a simple TPFDD problem:

1. *Consistent POE and POD* – The POE and POD of each movement requirement on a given trip of a resource must be the same.
2. *Consistent Resource Class* – Each resource can handle only some movement types. For example, a C-141 can handle bulk and oversize movements, but not outsize movements.
3. *Consistent PAX and Cargo Capacity* – The capacity of each resource cannot be exceeded.
4. *Consistent Release Time* – The start time of a movement (its Available to Load Date (ALD)) must not precede its release time.
5. *Consistent Arrival time* – The finish time of a trip must not precede the Earliest Arrival Date (EAD) of any of the transported movement requirements.
6. *Consistent Due time* – The finish time of a movement (its Latest Arrival Date (LAD)) must not be later than its due time.
7. *Consistent Trip Separation* – Movements scheduled on the same resource must start either simultaneously or with enough separation to allow for return trips. The inherently disjunctive and relative nature of this constraint makes it more difficult to satisfy than the others.

8. *Consistent Resource Use* – Only the given resources are used.

9. *Completeness* – All movement requirements must be scheduled.

In the next section we discuss the formalization of the above concepts. This problem does not consider certain aspects of transportation scheduling, such as aircrew scheduling, ground crew scheduling, maintenance, resource utilization rates, load/unload rates, port characteristics, etc. Each of these problem features have been handled in various more elaborate specifications.

5.3 (Re-)Formulating Domain Theories for Transportation Scheduling

In the most general view, scheduling is the construction of a set of reservations that satisfy given feasibility constraints and achieve "good" values of an objective function. Formally, the schedule is a relation, or even a simple relational database. A formal domain theory based on this view is given in Appendix A in [41]. The theory provides precise definitions for the concepts, constraints, objectives, and laws used to model this application domain.

This relational view however is not always the most efficient for particular problems. We may be able to reformulate the problem, incorporating constraints and objectives, yielding a problem statement that is more amenable to efficient problem-solving. In the following we present a series of transformations that reformulate the domain theory.

In most transportation problems, each movement requirement corresponds to a unique reservation – it is scheduled exactly once with a unique resource and start time. We can make this functional dependence explicit by treating a schedule as a *map* from movement requirements to resource/time tuples. In Figure 2 we show the effect of this reformulation on the schedule datatype.

Next the trip separation constraint suggests that this map is many-to-one, since several movements can take place simultaneously on the same resource. Inverting the map will induxe a partition on movement requirements. In terms of the transportation domain, inverting the map will make simultaneous movements explicit and thereby introducing the concept of a *trip* and the *manifest* of a trip.

Next we notice that the domain of a schedule map is a product of two types and these types have quite different properties (algebras): resources are a discrete set and time is (effectively) continuous and linear. The linear nature of time can be exploited by currying (to separate the two domain datatypes) and transforming the submap (from time to manifest) to a sequence, thereby making the linear structure of time explicit and introducing the concept of an *itinerary*.

This series of reformulations has dramatic effect on the trip separation constraint. In the initial formulation (in terms of reservations) this constraint involves $O(n^2)$ binary constraints between the n movements scheduled on a given resource. In the final formulation (in terms

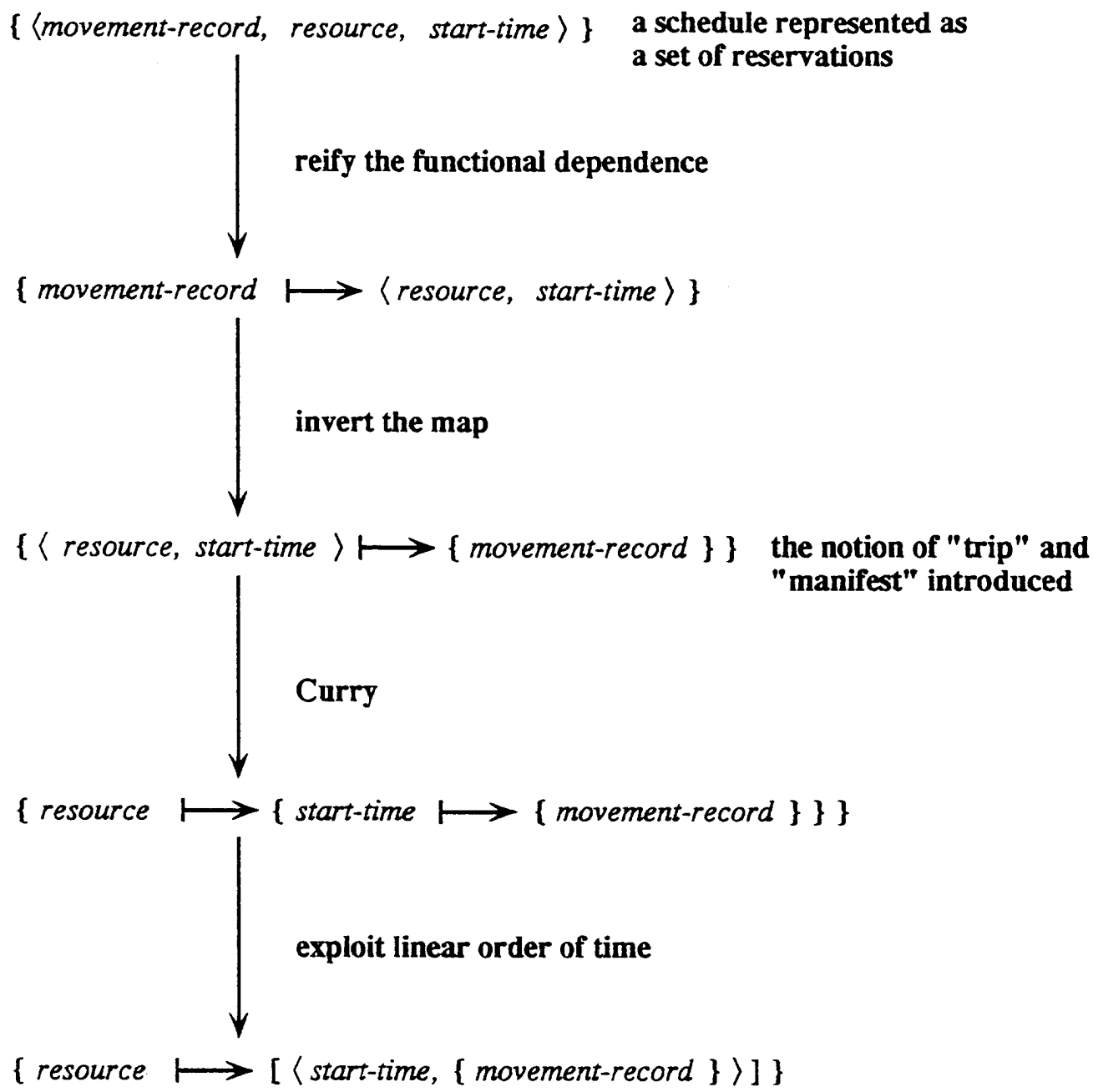


Figure 2: Reformulating a Scheduling Specification

of a linearized inverse map) this constraint is reduced to $O(n)$ binary constraints between the start times of consecutive trips.

For example, on a transportation problem involving over 15,000 movement requirements obtained from the U.S. Transportation Command, the scheduler produces a complete feasible schedule in about five minutes. A straightforward constraint network formulation based on this problem data would have over 31,000 variables and 120,125,000 constraints. Incorporating some of the structure of the problem, such as the linearity of time, allows reformulating this to a system of about 108,700 constraints. However, this is still a such large formulation that it seems an implicit representation is necessary to find feasible schedules efficiently.

The final reformulation is given in Appendix B in [41] and is the theory actually used to derive a scheduler.

5.4 Formal Specification of a Scheduler

The informal specification above can be expressed as follows:

function *TS*
  (*mvrs* : *seq*(*movement-record*),
  assets : *seq*(*resource-name*))
returns (*sched* : *map*(*resource-name*, *seq*(*trip*)) |
    Consistent-POE(*sched*)
    ∧ *Consistent-POD*(*sched*)
    ∧ *Consistent-Release-Times*(*sched*)
    ∧ *Consistent-Arrival-Times*(*sched*)
    ∧ *Consistent-Due-Times*(*sched*)
    ∧ *Consistent-Trip-Separation*(*sched*)
    ∧ *Consistent-Pax-Resource-Capacity*(*sched*)
    ∧ *Consistent-Cargo-Resource-Capacity*(*sched*)
    ∧ *Consistent-Movement-Type-and-Resource*(*sched*)
    ∧ *Available-Resources-Used*(*assets*, *sched*)
    ∧ *Scheduled-mvrs*(*sched*) = *seq-to-set*(*mvrs*))

This specifies a function called *TS* that takes two inputs, a sequence of movement records called *mvrs* and a sequence of resources called *assets*. The function returns a schedule, which has type *map*(*resource-name*, *seq*(*trip*)) and must satisfy the 11 conjoined constraints. Each constraint is defined in the domain theory; for example:

function *CONSISTENT-DUE-TIMES*
  (*sched* : *schedule*) : *boolean*
  = ∀(*rsrc* : *resource-name*, *trp* : *integer*, *mvr* : *movement-record*)
      (*rsrc* ∈ *domain*(*sched*)

$$
\begin{array}{l}
\wedge\ trp \in [1..size(sched(rsrc))] \\
\wedge\ mvr \in sched(rsrc)(trp).manifest \\
\Longrightarrow \\
sched(rsrc)(trp).start\text{-}time \\
\leq (mvr.due\text{-}date - sched(rsrc)(trp).trip\text{-}duration)
\end{array}
$$

This predicate expresses the constraint that every scheduled movement-record arrives before its due date.

6 Synthesizing a Scheduler

6.1 Approach

6.1.1 Problem Theories

We briefly review some basic concepts from algebra and logic. A *theory* is a structure $\langle S, \Sigma, A \rangle$ consisting of a set of sort symbols S, operations over those sorts Σ, and axioms A to constrain the meaning of the operations. A *theory morphism* (*theory interpretation*) maps from the sorts and operations of one theory to the sorts and expressions over the operations of another theory such that the image of each source theory axiom is valid in the target theory. A *parameterized theory* has formal parameters that are themselves theories [15]. The binding of actual values to formal parameters is accomplished by a theory morphism. Theory $\mathcal{T}_2 = \langle S_2, \Sigma_2, A_2 \rangle$ *extends* (or is an *extension* of) theory $\mathcal{T}_1 = \langle S_1, \Sigma_1, A_1 \rangle$ if $S_1 \subseteq S_2$, $\Sigma_1 \subseteq \Sigma_2$, and $A_1 \subseteq A_2$.

Problem theories define a problem by specifying a domain of problem instances or inputs and the notion of what constitutes a solution to a given problem instance. Formally, a *problem theory* $\mathcal{B}$ has the following structure.

Sorts	D, R
Operations	$I : D \rightarrow Boolean$
	$O : D \times R \rightarrow Boolean$

The *input condition* $I(x)$ constrains the input domain D. The *output condition* $O(x, z)$ describes the conditions under which output domain value $z \in R$ is a *feasible solution* with respect to input $x \in D$. Theories of booleans and sets are implicitly imported. Problems of finding optimal feasible solutions can be treated as extensions of problem theory by adding a cost domain, cost function, and ordering on the cost domain.

For example, the problem of finding feasible schedules can be presented as a problem theory via a theory interpretation into the domain theory of transportation scheduling:[1]

$$
\begin{array}{lcl}
D & \mapsto & seq(movement\text{–}record) \times seq(resource) \\
I & \mapsto & \lambda(Mvrs, resources)\ true \\
R & \mapsto & map(resource, seq(trip)) \\
O & \mapsto & \lambda(Mvrs, resources, sched) \\
 & & \quad Consistent\text{–}POE(sched) \\
 & & \quad \wedge\ Consistent\text{–}POD(sched) \\
 & & \quad \wedge\ Consistent\text{–}Release\text{–}Times(sched) \\
 & & \quad \wedge\ Consistent\text{–}Arrival\text{–}Times(sched) \\
 & & \quad \wedge\ Consistent\text{–}Due\text{–}Times(sched) \\
 & & \quad \wedge\ Consistent\text{–}Trip\text{–}Separation(sched) \\
 & & \quad \wedge\ Consistent\text{–}Pax\text{–}Resource\text{–}Capacity(sched) \\
 & & \quad \wedge\ Consistent\text{–}Cargo\text{–}Resource\text{–}Capacity(sched) \\
 & & \quad \wedge\ Consistent\text{–}Movement\text{–}Type\text{–}and\text{–}Resource(sched) \\
 & & \quad \wedge\ Available\text{–}Resources\text{–}Used(resources, sched) \\
 & & \quad \wedge\ Scheduled\text{–}mvrs(sched) = seq\text{–}to\text{–}set(mvrs)
\end{array}
$$

6.1.2 Algorithm Theories

An *algorithm theory* represents the essential structure of a certain class of algorithms A [43]. Algorithm theory $\mathcal{A}$ extends problem theory $\mathcal{B}$ with any additional sorts, operators, and axioms needed to support the correct construction of an $\mathcal{A}$ algorithm for $\mathcal{B}$. A theory morphism from the algorithm theory into some problem domain theory provides the problem-specific concepts needed to construct an instance of an $\mathcal{A}$ algorithm.

For example, global search theory (presented below in Section 6.2.1) extends problem theory with the basic concepts of backtracking: subspace descriptors, initial space, the splitting and extraction operations, filters, and so on. A divide-and-conquer theory would extend problem theory with concepts such as decomposition operators and composition operators [33, 36].

6.2 Synthesizing a Scheduler

There are two basic approaches to computing a schedule: local and global. Local methods focus on individual schedules and similarity relationships between them. Once an initial schedule is obtained, it is iteratively improved by moving to neighboring structurally similar schedules. Repair strategies [53, 25, 4, 31], and fixpoint iteration [8], and linear programming algorithms are examples of local methods.

Global methods focus on sets of schedules. A feasible or optimal schedule is found by repeatedly splitting an initial set of schedules into subsets until a feasible or optimal schedule

[1]The domain theory includes definitions for the types of movement-record, resource, trip (a record comprised of start-time and manifest), and schedule (a map from resource to sequence of trip).

can be easily extracted. Backtrack, heuristic search, and branch-and-bound methods are all examples of global methods. We explore the application of global methods. In the following subsections we formalize the notion of global search method and show how it can be applied to synthesize a scheduler. Other projects taking a global approach include ISIS [13], OPIS/DITOPS [47], and MicroBoss [28] (all at CMU).

6.2.1 Global Search Theory

The basic idea of global search is to represent and manipulate sets of candidate solutions. The principal operations are to *extract* candidate solutions from a set and to *split* a set into subsets. Derived operations include various *filters* which are used to eliminate sets containing no feasible or optimal solutions. Global search algorithms work as follows: starting from an initial set that contains all solutions to the given problem instance, the algorithm repeatedly extracts solutions, splits sets, and eliminates sets via filters until no sets remain to be split. The process is often described as a tree (or DAG) search in which a node represents a set of candidates and an arc represents the split relationship between set and subset. The filters serve to prune off branches of the tree that cannot lead to solutions.

The sets of candidate solutions are often infinite and even when finite they are rarely represented extensionally. Thus global search algorithms are based on an abstract data type of intensional representations called *space descriptors* (denoted by hatted symbols). In addition to the extraction and splitting operations mentioned above, the type also includes a predicate *satisfies* that determines when a candidate solution is in the set denoted by a descriptor. Further, there is a refinement relation on spaces that corresponds to the subset relation on the sets denoted by a pair of descriptors.

The various operations in the abstract data type of space descriptors together with problem specification can be packaged together as a theory. Formally, abstract *global search theory* (or simply *gs–theory*) $\mathcal{G}$ is presented in Figure 3, where D is the input domain, R is the output domain, I is the input condition, O is the output condition, $\hat{R}$ is the type of space descriptors, $\hat{I}$ defines legal space descriptors, $\hat{r}$ and $\hat{s}$ vary over descriptors, $top(x)$ is the descriptor of the initial set of candidate solutions, $Satisfies(z, \hat{r})$ means that z is in the set denoted by descriptor $\hat{r}$ or that z satisfies the constraints that $\hat{r}$ represents, and $Extract(z, \hat{r})$ means that z is directly extractable from $\hat{r}$.

The relations *Split–Arg* and *Split–Constraint* are used to determine and perform splitting. In particular, if $Split\text{–}Arg(x, \hat{r}, c)$ then c is information that characterizes (or informs) one branch of the split. $Split\text{–}Constraint(x, \hat{r}, c, \hat{s})$ means that $\hat{s}$ results from incorporating information c into the descriptor $\hat{r}$ (with respect to input x). *Split–Arg* is used to control the generation of children of a node in the search tree and *Split–Constraint* is used to specify one child. *Split–Constraint* can be thought of as a parameterized constraint whose alternative arguments are supplied by *Split–Arg*.

The refinement relation $\hat{r} \sqsupseteq \hat{s}$ holds when $\hat{s}$ denotes a subset of the set denoted by $\hat{r}$. Further, $\hat{R}$ together with $\sqsupseteq$ forms a bounded semilattice. This structure will play a crucial role in constraint propagation algorithms.

Spec *Global-Search*

Sorts

	D	*input domain*
	R	*output domain*
	$\hat{R}$	*subspace descriptors*
	$\hat{C}$	*splitting information*

Operations

$I : D \rightarrow boolean$	*input condition*
$O : D \times R \rightarrow boolean$	*input/output condition*
$\hat{I} : D \times \hat{R} \rightarrow boolean$	*subspace descriptors condition*
$Satisfies : R \times \hat{R} \rightarrow boolean$	*denotation of descriptors*
$Split\text{–}Arg : D \times \hat{C} \times \hat{R} \rightarrow boolean$	*specifies arguments to split constraint*
$Split\text{–}Constraint : D \times \hat{R} \times \hat{C} \times \hat{R} \rightarrow boolean$	*parameterized splitting constraint*
$Extract : R \times \hat{R} \rightarrow boolean$	*extractor of solutions from spaces*
$\Psi : D \times R \times \hat{R} \rightarrow boolean$	*cutting constraint*
$\xi : D \times \hat{R} \rightarrow boolean$	*cutting constraint*
$\sqsupseteq : D \times \hat{R} \times \hat{R} \rightarrow boolean$	*refinement relation*
$top : D \rightarrow \hat{R}$	*initial space*
$bot : \hat{R}$	*inconsistent space*

Axioms

GS0. All feasible solutions are in the *top* space

$$I(x) \wedge O(x,z) \Longrightarrow Satisfies(z, top(x))$$

GS1. All solutions in a space are finitely extractable

$$I(x) \wedge \hat{I}(x,\hat{r}) \Longrightarrow (Satisfies(z,\hat{r}) \Longleftrightarrow \exists(\hat{s})\ (\ Split^*(x,\hat{r},\hat{s}) \wedge Extract(z,\hat{s})))$$

GS2. Specification of Cutting Constraint

$$Satisfies(z,\hat{r}) \wedge O(x,z) \Longrightarrow \Psi(x,z,\hat{r})$$

GS3. Definition of Cutting Constraint on Spaces

$$\xi(x,\hat{r}) \Longleftrightarrow \forall(z:R)(\ Sat(z,\hat{r}) \Longrightarrow \Psi(x,z,\hat{r}))$$

GS4. Definition of Refinement

$$\hat{r} \sqsupseteq \hat{s} \Longleftrightarrow \forall(z:R)(Satisfies(z,\hat{s}) \Longrightarrow Satisfies(z,\hat{r}))$$

GS5. $\langle \hat{R}, \sqsupseteq, \sqcap, top, bot \rangle$ is a bounded meet-semilattice with *bot* as universal lower bound.

end spec

Figure 3: Global Search Theory

Note that all variables in the axioms are assumed to be universally quantified unless explicitly specified otherwise. Axiom GS0 asserts that the initial descriptor $\hat{r}_0(x)$ is a legal descriptor. Axiom GS1 asserts that legal descriptors split into legal descriptors and that *Split* induces a well-founded ordering on spaces. Axiom GS2 constrains the denotation of the initial descriptor — all feasible solutions are contained in the initial space. Axiom GS3 gives the denotation of an arbitrary descriptor $\hat{r}$ — an output object z is in the set denoted by $\hat{r}$ if and only if z can be extracted after finitely many applications of *Split* to $\hat{r}$ where

$$Split^*(x, \hat{r}, \hat{s}) \iff \exists(k : Nat)\ \ Split^k(x, \hat{r}, \hat{s})$$

and

$$Split^0(x, \hat{r}, \hat{t}) \iff \hat{r} = \hat{t}$$

and for all natural numbers k

$$Split^{k+1}(x, \hat{r}, \hat{t})$$
$$\iff \exists(\hat{s} : \hat{R},\ i : \hat{C})\ (\ Split\text{–}Arg(x, \hat{r}, i) \wedge Split\text{–}Constraint(x, \hat{r}, i, \hat{s}) \wedge Split^k(x, \hat{s}, \hat{t})).$$

Axiom GS4 asserts that if $\hat{r}$ splits to $\hat{s}$ then $\hat{r}$ also refines to $\hat{s}$; thus the refinement relation on $\hat{R}$ is weaker than the split relation. We also need the axioms that $\langle \hat{R}, \sqsupseteq, \sqcap \rangle$ is a semilattice. For simplicity, we write $\hat{r} \sqsupseteq \hat{s}$ rather than the correct $\sqsupseteq (x, \hat{r}, \hat{s})$; and similarly $\hat{r} \sqcap \hat{s}$.

For example, a simple global search theory of scheduling has the following form. Schedules are represented as maps from resources to sequences of trips, where each trip includes earliest-start-time, latest-start-time, port of embarkation, port of debarkation, and manifest (set of movement records or ULNs + CINs + PINs from the TPFDD). The type of schedules has the invariant (or subtype characteristic) that for each trip, the earliest-start-time is no later than the latest-start-time. A partial schedule is a schedule over a subset of the given movement records.

The initial (partial) schedule is just the empty schedule – a map from the available resources to the empty sequence of trips. A partial schedule is extended by first selecting a movement record *mvr* to schedule, then selecting a resource r, and then a trip t on r (either an existing trip or a newly created one) – the triple $\langle mvr, r, t \rangle$ constitutes the information c of *Split–Arg*. *Split–Constraint* given $\langle mvr, r, t \rangle$ creates an extended schedule that has *mvr* added to the manifest of trip t on resource r. The alternative ways that a partial schedule can be extended naturally gives rise to the branching structure underlying global search algorithms.

The formal version of this global search theory of scheduling can be inspected in the domain theory in Appendix B in [41].

6.2.2 Pruning Mechanisms

When a partial schedule is extended it is possible that some problem constraints are violated in such a way that further extension to a complete feasible schedule is impossible. In tree search algorithms it is crucial to detect such violations as early as possible.

Pruning tests are derived in the following way. The test

$$\exists(z)\ (\mathit{Satisfies}(z, \hat{r})\ \wedge\ O(x, z)) \tag{1}$$

decides whether there exist any feasible solutions that are in the space denoted by $\hat{r}$. If we could decide this at each node of our branching structure then we would have perfect search – no deadend branches would ever be explored. In practice it would be impossible or horribly complex to compute (1), so we rely instead on an inexpensive approximation to it. In fact, if we approximate (1) by weakening it (deriving a necessary condition of it) we obtain a sound pruning test. That is, suppose we can derive a test $\Phi(x, \hat{r})$ such that

$$\exists(\mathit{sched})\ (\mathit{Satisfies}(z, \hat{r})\ \wedge\ O(x, z)) \implies \Phi(x, \hat{r}). \tag{2}$$

By the contrapositive of (2), if $\neg\Phi(x, \hat{r})$ then there are no feasible solutions in $\hat{r}$, so we can eliminate it from further processing. A global search algorithm will test Φ at each node it explores, pruning those nodes where the test fails.

More generally, necessary conditions on the existence of feasible (or optimal) solutions below a node in a branching structure underlie pruning in backtracking and the bounding and dominance tests of branch-and-bound algorithms [34].

It appears that the bottleneck analysis advocated in the constraint-directed search projects at CMU [12, 28] leads to a semantic approximation to (1), but neither a necessary nor sufficient condition. Such a *heuristic* evaluation of a node is inherently fallible, but if the approximation is close enough it can provide good search control with relatively little backtracking.

To derive pruning tests for the strategic transportation scheduling problem, we instantiate (1) with our definition of *Satisfies* and O and use an inference system to derive necessary conditions. The resulting tests are fairly straightforward; of the 11 original feasibility constraints, 7 yield pruning tests on partial schedules. For example, the partial schedule must satisfy *Consistent-POE*, *Consistent-POD*, *Consistent-Pax-Resource-Capacity*, *Consistent-Cargo-Resource-Capacity*, *Consistent-Movement-Type-and-Resource*, and *Available-Resources-Used*. The reader may note that computing these tests on partial schedules is rather expensive and mostly unnecessary – later program optimization steps will however reduce these tests to fast and irredundant form. For example, the first test will reduce to checking that when we place a movement record *mvr* on trip t, we check that the POE of *mvr* and t are consistent.

For details of deriving pruning mechanisms for other problems see [34, 42, 43, 35].

6.2.3 Cutting Constraints and Constraint Propagation

Constraint propagation is a more general technique that is crucial for early detection of infeasibility. We developed a general mechanism for deriving constraint propagation code and applied it to scheduling.

Each node in a backtrack tree can be viewed as a data structure that denotes a set of candidate solutions – in particular the solutions that occur in the subtree rooted at the node (see Figure 4). Thus the root denotes the set of all candidate solutions found in the tree.

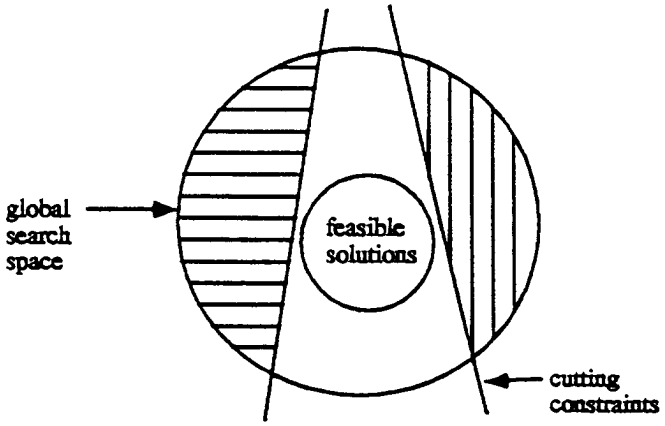


Figure 4: Global Search Subspace and Cutting Constraints

Pruning has the effect of removing a node (set of solutions) from further consideration. In contrast, constraint propagation has the effect of changing the space descriptor so that it denotes a smaller set of candidate solutions. The effect of constraint propagation is to spread information through the subspace descriptor resulting in a tighter descriptor and possibly exposing infeasibility. Pruning can be treated as a special case of propagation in which a space is refined to descriptor that denotes the empty set of solutions.

Constraint propagation is based on the notion of *cutting constraints* which are necessary conditions $\Psi(x, z, \hat{r})$ that a candidate solution z satisfying $\hat{r}$ is feasible:

$$\forall(x : D, \hat{r} : \hat{R},\ z : R)(\mathit{Satisfies}(z, \hat{r}) \ \wedge\ O(x, z) \implies \Psi(x, z, \hat{r})) \tag{3}$$

See Figures 4 and 5. In order to get a test on spaces that decides whether Ψ has been incorporated, we make one further definition:

$$\xi(x, \hat{r}) \iff \forall(z : R)(\mathit{Satisfies}(z, \hat{r}) \implies \Psi(x, z, \hat{r})) \tag{4}$$

The test $\xi(x, \hat{r})$ holds exactly when all candidate solutions in $\hat{r}$ satisfy Ψ, and we say that $\hat{r}$ *satisfies* ξ.

The key question at this point is: Given a descriptor $\hat{r}$ that doesn't satisfy ξ, how can we incorporate ξ into $\hat{r}$? The answer is to find the greatest refinement of $\hat{r}$ that satisfies ξ; we say $\hat{t}$ *incorporates* ξ into $\hat{r}$ if

$$\hat{t} = max_{\sqsupseteq}\{\hat{s} \mid \hat{r} \sqsupseteq \hat{s} \ \wedge\ \xi(x, \hat{s})\}. \tag{5}$$

which asserts that $\hat{t}$ is maximal over the set of descriptors that refine $\hat{s}$ and satisfy ξ, with respect to ordering $\sqsupseteq$. We want $\hat{t}$ to be a refinement of $\hat{r}$ so that all of the information in

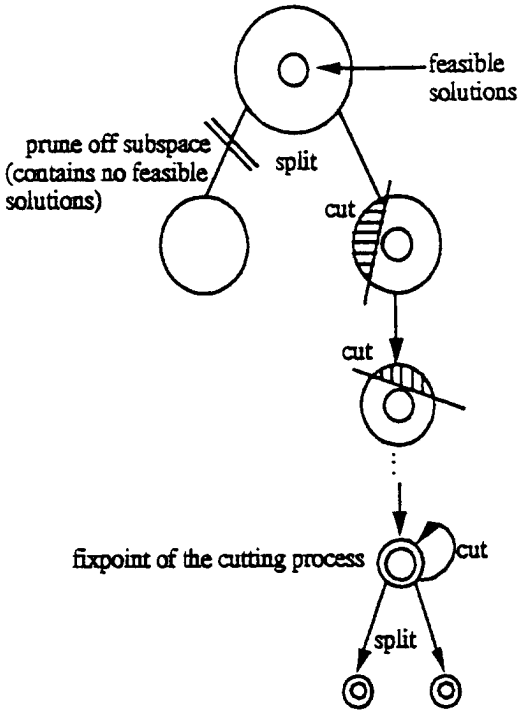


Figure 5: Pruning and Constraint Propagation

$\hat{r}$ is preserved and we want $\hat{t}$ to be maximal so that no other information than $\hat{r}$ and ξ is incorporated into $\hat{t}$.

The next question concerns the conditions under which Formula (5) is satisfiable. Assuming that $\hat{R}$ is a semilattice, we can use variants of Tarski's fixpoint theorem (c.f. [8]):

Theorem If there is a function f such that

1. f is monotonic on $\hat{R}$ (i.e. $\hat{s} \sqsupseteq \hat{t} \implies f(x,\hat{s}) \sqsupseteq f(x,\hat{t})$)
2. f is deflationary (i.e. $\hat{r} \sqsupseteq f(x,\hat{r})$)
3. f has fixed-points satisfying ξ (i.e. $f(x,\hat{r}) = \hat{r} \iff \xi(x,\hat{r})$)

then (1) $\hat{t} = max_{\sqsupseteq}\{\hat{s} \mid \hat{r} \sqsupseteq \hat{s} \wedge \xi(x,\hat{s})\}$ exists
and (2) $\hat{t}$ is the greatest fixpoint of f; i.e. $\hat{t}$ can be computed by iteratively applying f to $\hat{r}$ until a fixpoint is reached.

The challenge is to construct a monotonic, deflationary function whose fixed-points satisfy ξ. A general construction in terms of global search theory can be sketched as follows. Let

$$f(x,\hat{r}) = \begin{cases} \hat{r} & \text{if } \xi(x,\hat{r}) \\ \ldots & \text{if } \neg\xi(x,\hat{r}) \end{cases}$$

The intent is to define f so that it has fixpoints exactly when $\xi(x, \hat{r})$ holds. When $\xi(x, \hat{r})$ doesn't hold, then we know (by the definition of ξ and the contrapositive of formula (3)) that

$$\exists(z : R)(\mathit{Satisfies}(z, \hat{r}) \ \wedge \ \neg O(x, z))$$

i.e. there are some infeasible solutions in the space described by $\hat{r}$. Ideally $\neg\xi(x, \hat{r})$ is a *constructive* assertion, so it provides information on *which* solutions are infeasible and how to eliminate them. In place of the ellipsis above we require a new descriptor that refines $\hat{r}$ (so f is decreasing on all inputs), allows f to be monotone, and eliminates some of the infeasible solutions indicated by $\neg\xi(x, \hat{r})$. In general it is difficult to see how to achieve this end without assuming special structure to $\hat{R}$ and ξ.

We have identified some special cases for which an analytic procedure can produce the necessary iteration function f from ξ. These special cases subsume our scheduling applications and many related Constraint Satisfaction Problems (CSP) problems. Suppose that the constraint ξ has the form

$$B(x, \hat{r}) \sqsupseteq \hat{r} \tag{6}$$

where $B(x, \hat{r})$ is monotonic in $\hat{r}$. We say that ξ is a *Horn-like constraint* by generalization of Horn clauses in logic. Notice that the occurrence of $\hat{r}$ on the right-hand side of the inequality has positive polarity (i.e. it is monotonic in $\hat{r}$), whereas the occurrence(s) of $\hat{r}$ on the left-hand side have negative polarity (i.e. are antimonotonic). If the constraint were boolean (with B and $\hat{r}$ being boolean values and $\sqsupseteq$ being implication), then this would be called a definite Horn clause. When our constraints are Horn-like, then there is a simple definition for the desired function f:

$$f(x, \hat{r}) = \begin{cases} \hat{r} & \text{if } B(x, \hat{r}) \sqsupseteq \hat{r} \\ B(x, \hat{r}) \sqcap \hat{r} & \text{if } \neg B(x, \hat{r}) \sqsupseteq \hat{r} \end{cases}$$

or equivalently

$$f(x, \hat{r}) = B(x, \hat{r}) \sqcap \hat{r}.$$

It is easy to check that f is monotone in $\hat{r}$, deflationary, and has fixed-points exactly when ξ holds. Therefore, simple iteration of f will converge to the descriptor that incorporates ξ into $\hat{r}$. However, if $\hat{r}$ is an aggregate structure such as a tuple or map, then the changes made at each iteration may be relatively sparse, so the simple iteration approach may be grossly inefficient. We found this feature to be characteristic of scheduling and other CSPs. Our approach to solving this problem is to focus on single point changes and to exploit dependence analysis. For each component of $\hat{r}$ we define a separate change propagation procedure. The arguments to a propagation procedure specify a change to the component. This change is performed and then the change procedures for all other components that could be affected by the change are invoked. Static dependence analysis at design-time is used to determine which constraints could be affected by a change to a given component.

A program scheme for global search with constraint propagation is presented in Figure 6. The global search design tactic in KIDS first instantiates this scheme, then invokes a tactic for synthesizing propagation code to satisfy the specification *F–split–and–propagate*.

Spec *Global-Search-Program (T :: Global-Search)*

Operations

F-initial-propagate $(x : D \mid I(x))$
returns $(\hat{t} : \hat{R} \mid \hat{t} = max_{\sqsupseteq} \{\hat{s} \mid top(x) \sqsupseteq \hat{s} \wedge \hat{I}(x, \hat{s}) \wedge \xi(x, \hat{s})\})$

F-split-and-propagate
$(x : D,\ \hat{r} : \hat{R},\ c : \hat{C}$
$\mid I(x) \wedge \hat{I}(x, \hat{r}) \wedge \mathit{Split\text{–}Arg}(x, \hat{r}, c) \wedge \xi(x, \hat{r}) \wedge \hat{r} \neq bot)$
returns $(\hat{t} : \hat{R} \mid \hat{t} = max_{\sqsupseteq} \{\hat{s} \mid \hat{r} \sqsupseteq \hat{s} \wedge \hat{I}(x, \hat{s})$
$\wedge\ \mathit{Split}(x, \hat{r}, c, \hat{s}) \wedge \xi(x, \hat{s})\})$

F-gs $(x : D,\ \hat{r} : \hat{R} \mid I(x) \wedge \hat{I}(x, \hat{r}) \wedge \Phi(x, \hat{r}))$
returns $(z : R \mid O(x, z) \wedge \mathit{Satisfies}(z, \hat{r}))$
$=$ **if** $\exists(z)\,(\mathit{Extract}(z, \hat{r}) \wedge O(x, z))$
then $\mathit{some}(z)\,(\mathit{Extract}(z, \hat{r}) \wedge O(x, z))$
else $\mathit{some}(z)\,\exists(c : \hat{C},\ \hat{t} : \hat{R})$
$(\mathit{Split\text{–}Arg}(x, \hat{r}, c)$
$\wedge\ \hat{t} = \mathit{F\text{-}split\text{-}and\text{-}propagate}(x, \hat{r}, c) \wedge \hat{t} \neq bot$
$\wedge\ z = \mathit{F\text{-}gs}(x, \hat{t}))$

$F\ (x : D \mid I(x))$
returns $(z : R \mid O(x, z))$
$= \mathit{some}(z)\,\exists(\hat{t})\ (\hat{t} = \mathit{F\text{-}initial\text{-}propagate}(x)$
$\wedge\ \hat{t} \neq bot$
$\wedge\ z = \mathit{F\text{-}gs}(x, \hat{t}))$

end spec

Figure 6: Global Search Program Theory

CSPs with Horn-like constraints

We now elaborate the previous exposition of propagation of Horn-like constraints arising in CSPs. To keep matters simple, yet general, suppose that the output datatype R is *map(VAR, VALSET)*, where VAR is a type of variables, and $VALSET$ is a type that denotes a set of values (this implies that all the variables have the same type and refinement ordering), and the $\sqsupseteq$ relation has the form:

$$\hat{r} \sqsupseteq \hat{s} \;\; \textit{iff} \;\; \bigwedge_v \hat{r}(v) \sqsupseteq \hat{s}(v).$$

Suppose further that ξ is a conjunction of constraints giving bounds on the variables:

$$\xi(x, \hat{r}) \iff \bigwedge_v B_v(x, \hat{r}) \sqsupseteq \hat{r}(v)$$

where $B_v(x, \hat{r})$ is monotonic in $\hat{r}$. Under these assumptions, $\neg\xi(x, \hat{r})$ implies that the bounding constraint on some variable v is violated; i.e.

$$\neg B_v(x, \hat{r}) \sqsupseteq \hat{r}(v).$$

To "fix" such a violation we can change the current valset of v to

$$B_v(x, \hat{r}) \sqcap \hat{r}(v),$$

which simultaneously refines $\hat{r}(v)$, since

$$\hat{r}(v) \sqsupseteq B_v(x, \hat{r}) \sqcap \hat{r}(v)$$

and reestablishes the constraint on v, since

$$B_v(x, \hat{r}) \sqsupseteq B_v(x, \hat{r}) \sqcap \hat{r}(v).$$

Let

$$B(x, \hat{r}) = \{\mid u \rightarrow B_u(x, \hat{r}) \sqcap \hat{r}(u) \mid u \in \textit{domain}(\hat{r}) \mid\}$$

then, define f as:

$$f(x, \hat{r}) = \hat{r} \sqcap B(x, \hat{r})$$

Constraint propagation is treated here as iteration of f until a fixed-point is reached. Efficiency requires that we go farther, since only a sparse subset of the variables in $\hat{r}$ will be updated at each iteration. If we implemented the iteration on a vector processor or SIMD machine, the overall computation could be fast, but wasteful of processors. On a sequential machine, it is advantageous to analyze the constraints in ξ to infer dependence of constraints on variables. That is, if (the valset of) variable v changes, which constraints in ξ could become violated? This dependence analysis can be used to generate special-purpose propagation code as follows.

For each variable v, let *affects(v)* be the set of variables whose constraints could be violated by a change in v; more formally, let

$$\textit{affects}(v) \;=\; \{u \mid v \text{ occurs in } B_u\}.$$

We can then generate a set of procedures that carry out the propagation/iteration of f: For each variable v, generate the following propagation procedure:

$Propagate_v\ (x : D,\ \hat{r} : \hat{R},\ new\text{–}valset : VALSET$
$\quad | I(x)\ \wedge\ \hat{I}(x, \hat{r})$
$\quad \wedge\ \hat{r}(v) \sqsupset new\text{–}valset$
$\quad \wedge\ B_v(x, \hat{r}) \sqsupseteq new\text{–}valset)$
$=$ **let** $(\hat{s} : \hat{R} = map\text{–}shadow(\hat{r}, v, new\text{–}valset))$
if $\neg\hat{I}(x, \hat{s})$ **then** *bot*
else
... for each variable u in $affects(v)$ ...
... generate the following code block ...
if $\hat{s}$ $=$ *bot* **then** *bot*
else (**if** $\neg(B_u(x, \hat{s}) \sqsupseteq \hat{s}(u))$
then $\hat{s} \leftarrow Propagate_u(x, \hat{s},\ B_u(x, \hat{s}) \sqcap \hat{s}(u)))$;
...
$\hat{s}$
end

where $map\text{–}shadow(\hat{r}, v, new\text{–}valset)$ returns the map $\hat{r}$ modified so that $\hat{r}(v) = new\text{–}valset$.

To finish up, if $Split(x, \hat{r}, i, \hat{s})$ has the form

$$\hat{s}(u) = C(x, \hat{r}, i)$$

for some function C that yields a refined valset for variable u, then we can satisfy *F–split–and–propagate* as follows:

$$F\text{–}split\text{–}and\text{–}propagate(x, \hat{r}, i) = propagate_u(\hat{r}, C(x, \hat{r}, i)).$$

The change to u induced in the call to $propagate_u$ will in turn trigger changes to other variables, and so on.

We have described the generation of constraint propagation in a relatively simple setting. One of the authors (Westfold) was largely responsible for the development and implementation of this work. The implementation treats a much broader range of problem features than has been described above. Further elaborations include

1. Heterogeneous variables (and semilattice/refinement structure)

2. Multiple constraints on each variable

3. Indexed variables

4. Conditional constraints

5. Dynamic set of variables

6. Ordering of constraints in propagation procedures

There are many ways to implement constraint propagation, this being just one. Our approach is useful when the *affects* relation is relatively sparse, so special control code to follow dependences and fixing violations is efficient. An alternative approach is to reify *affects* via explicit links between variables, forming a constraint network. The synthesis of the propagation control strategy is relatively simple, since we only need to follow dependence links. Disadvantages of this approach include the size of the constraint network and the cost of maintaining it. This is a common approach in the CSP literature.

Our model of constraint propagation generalizes the concepts of cutting planes in the Operations Research literature [26] and the forms of propagation studied in the constraint satisfaction literature (e.g. [18]). Our use of fixed-point iteration for constraint propagation is similar to Paige's work on fixed-point iteration in RAPTS [8]. The main differences are (1) RAPTS expects the user to supply the monotone function as part of the initial specification whereas we derive it from a more abstract statement of the problem; (2) RAPTS instantiates a straightforward iteration scheme and then performs optimizations. Such an approach would be too inefficient for scheduling applications, so we use dependence analysis to generate code that is specific to the constraint system at hand.

Constraint Propagation for Transportation Scheduling

For transportation scheduling, each iteration of the *Propagate* operation has the following form, where est_i denotes the earliest-start-time for trip i and est'_i denotes the next value of the earliest-start-time for trip i (analogously, lst_i denotes latest-start-time), and $roundtrip_i$ is the roundtrip time for trip i on resource r. For each resource r and the i^{th} trip on r,

$$est'_i = max \begin{cases} est_i \\ est_{i-1} + roundtrip_i \\ max\text{–}release\text{–}time(manifest_i) \end{cases}$$

$$lst'_i = min \begin{cases} lst_i \\ lst_{i+1} - roundtrip_i \\ min\text{–}finish\text{–}time(manifest_i) \end{cases}$$

Here $max\text{–}release\text{–}time(manifest_i)$ computes the max over all of the release times of movement requirements in the manifest of trip i and $min\text{–}finish\text{–}time(manifest_i)$ computes the minimum of the finish times of movement requirements in the same manifest. Boundary cases must be handled appropriately.

After adding a new movement record to some trip, the effect of *Propagate* will be to shrink the

$$(earliest\text{–}start\text{–}time, latest\text{–}start\text{–}time)$$

window of each trip on the same resource. If the window becomes negative for any trip, then the partial schedule is necessarily infeasible and it can be pruned.

The constraint propagation code generated for TS in Appendix C in [41], nearly as fast as handwritten propagation code for the same problem (cf. Appendix C in [37]).

6.2.4 Constraint Relaxation

Many scheduling problems are overconstrained. Overconstrained problems are typically handled by relaxing the constraints. The usual method, known as Lagrangian Relaxation [26], is to move constraints into the objective function. This entails reformulating the constraint so that it yields a quantitative measure of how well it has been satisfied.

Another approach is to relax the input data just enough that a feasible solution exists. To test this approach, we hand-modified one version of KTS so it relaxes the LAD (Latest Arrival Date) constraint. The relaxation takes place only when there is no feasible solution to the problem data. KTS keeps track of a quantitative measure of each LAD violation (e.g. the difference between the arrival date of a trip and the LAD of a movement requirement in that trip). If there is no feasible reservation for the movement requirement being scheduled, then KTS uses the recorded information to relax its the LAD. The relaxation is such as to minimally delay the arrival of the requirement to its POD.

This technique, which we call *data relaxation*, can be described more generally. Suppose that we specify a certain constraint to be relaxable. Whenver we detect that the input data has no feasible solution, we attempt to relax the input data just enough to allow a feasible solution. Of course, the problem-solving process and data relaxation are interleaved.

At each global search iteration we evaluate this objective function for all candidate solutions. Using these values the algorithm takes a greedy decision of which branch of the global search tree should be split next. The result is a heuristically-guided algorithm that finds good but not necessarily optimal schedules.

It remains an open task to formalize the notion of data relaxation and to develop a tactic for synthesizing relaxation code in the context of global search with constraint propagation.

6.2.5 Using KIDS

In developing a new scheduling application, most of the user's time is spent building a theory of the domain. Our scheduling theories have evolved over months of effort into 50-70 pages of text. It currently takes about 90 minutes to transform our most complex scheduling specification (for ITAS) into optimized and compiled CommonLisp code for Sun workstations. Evolution of the scheduler is performed by evolving the domain theory and specification, followed by regeneration of code.

Currently, the global search deisgn tactic in KIDS is used to design an algorithm for F and $F{-}gs$ in Figuregs-scheme. A specialized tactic for generating constraint propagation

Data Sets (Air only)	# of input TPFDD records (ULNs)	# of individual movements	# of scheduled items after splitting	Solution time	Msec per scheduled item
CDART		296	330	0.5 sec	1.5
CSRT01	1,600	1,261	3,557	44 sec	12
096-KS	20,400	4,644	6,183	86 sec	14
9002T Borneo	28,900	10,623	15,119	290 sec	20

Figure 7: KTS Scheduling Statistics

code for Horn-like constraints is used to generate code for *F-split-and-propagate*. Once the algorithm is designed, then a series of simplification and common-subexpression-elimination transformations are applied. A trace of the KIDS derivation is given in Appendix C in [41]. See [42] for a detailed description of a session with KIDS.

6.3 KTS – Strategic Transportation Scheduling

The KTS schedulers synthesized using the KIDS program transformation system are extremely fast and accurate [44, 45]. The chart in Figure 7 lists 4 TPFDD problems, and for each problem (1) the number of TPFDD lines (each requirement line contains up to several hundred fields), (2) the number of individual movement requirements obtained from the TPFDD line (each line can specify several individual movements requirements), (3) the number of movement requirements obtained after splitting (some requirements are too large to fit on a single aircraft or ship so they must be split), (4) the cpu time to generate a complete schedule, and (5) time spent per scheduled movement. Similar results were obtained for sea movements. The largest problem, Borneo NEO, is harder to solve, because of the presence of 29 movement requirements that are inherently unschedulable: their due date comes before their availability date. Such inconsistencies must be expected and handled by a realistic system. KTS simply relaxes the due date the minimal amount necessary to obtain a feasible schedule.

We compared the performance of KTS with several other TPFDD scheduling systems: JFAST, FLOGEN, DITOPS, and PFE. We do not have access to JFAST and FLOGEN, but these are (or were) operational tools at AMC (Airlift Mobility Command, Scott AFB). According to [11] and David Brown (retired military planner consulting with the Planning Initiative), on a typical TPFDD of about 10,000 movement records, JFAST takes several hours and FLOGEN about 36 hours. KTS on a TPFDD of this size will produce a detailed

schedule in *one to three minutes.* So KTS seems to be a factor of about 25 times faster than JFAST and over 250 times faster than FLOGEN. The currently operational ADANS system reportedly runs at about the same speed as FLOGEN. When comparing schedulers it is also important to compare the transportation models that they support. KTS has a richer model than JFAST (i.e. handles more constraints and problem features), but less rich than ADANS. The ITAS effort described in the next section reflects our efforts to synthesize schedulers that have at least the richness of the ADANS model.

The DITOPS project at CMU also models scheduling as a constraint satisfaction problem. However, DITOPS effectively interprets its problem constraints, whereas the transformational approach can produce highly optimized "compiled" constraint operations. DITOPS emphasizes complex heuristics for guiding the search away from potential bottlenecks. In contrast KTS uses simple depth-first search but emphasizes the use of strong and extremely fast pruning and constraint propagation code. DITOPS requires minutes to solve the CDART data. KTS finds a complete feasible solution in 0.5 seconds.

Comparison with PFE (Prototype Feasibility Estimator, built by BBN based on the Transportation Feasibility Estimator system): On the MEDCOM-SITUATION from the CPE (Common Prototype Environment), KTS is about 5 times faster than PFE and produces a SEA schedule with only 14% of the delay of the PFE schedule. KTS also produces a far more accurate estimate of the planes needed to handle the AIR movements, since PFE is only estimating feasibility whereas KTS produces a detailed schedule.

In our Strategic TPFDD scheduler KTS, we explored issues of speed and embedding KTS into an easy-to-use GUI, complete with ability to edit the data model (TPFDD, resource classes and instances, and port models), to schedule, apply various analysis tools, and to dynamically reschedule. KTS is available from Kestrel via ftp to participants in the PI.

6.4 ITAS – In-Theater Airlift Scheduler

In 1994 we began to develop a scheduler to support PACAF (Pacific Air Force) at Hickham AFB, Honolulu which is tasked with in-theater scheduling of a fleet of 26 C-130 cargo aircraft in the Pacific region. Several variants of a theater scheduler, called ITAS for In-Theater Airlift Scheduler, have been developed to date, and more are planned. The system runs on laptop computers (Apple Powerbook). The interface to ITAS and integration with a commercial database package have been developed by BBN. Users enter information about movement requirements, available resources, port features, etc. and ITAS automatically generates a schedule, displayed in a gantt-like "rainbow" chart. The schedule can also be printed in the form of ATO's (Air Tasking Orders).

The ITAS schedulers have emphasized flexibility and rich constraint modeling. The version of ITAS installed at PACAF in February 1995 simultaneously schedules the following types of resources:

1. aircraft

2. aircrews and their duty day cycles
3. ground crews for unloading
4. parking space at ports

each of which may have a variety of attendant constraints and problem features.

7 Classification Approach to Algorithm Design

In this section we introduce a new knowledge-based approach to algorithm design. We have been developing it in order to support the incremental application of problem-solving methods to scheduling problems. Our techniques enable us to integrate at a deep semantic level problem-solving methods from Computer Science (e.g. divide-and-conquer, global search), Artificial Intelligence (e.g. heuristic search, constraint propagation, neural nets), and Operations Research (e.g. Simplex, integer programming, network algorithms). Furthermore these techniques have applications far wider than algorithm design, since they apply to the incremental application of *any* kind of knowledge formally represented in a hierarchy.

7.1 Technical Foundations – Theories

A *theory* (i.e. first-order theory presentation) defines a language and constrains the possible meanings of its symbols by axioms and inference rules. Theories can be used to express many kinds of software-related artifacts, including domain models [49], formal requirements [3, 10, 27, 29], programming languages [6, 15, 19], abstract data types and modules [10, 14, 17], and abstract algorithms [43]. There has been much work on operations for constructing larger theories from smaller theories [3, 7, 30].

A *theory morphism* translates the language of one theory into the language of another theory in a way that preserves theorems. Theory morphisms underlie several aspects of software development, including specification refinement and datatype implementation [5, 27, 30, 51], the binding of parameters in parameterized theories [9, 15], algorithm design [22, 43, 52], and data structure design [32]. There has been work on techniques for composing implementations in a way that reflects the structure of the source specification [3, 30]; however these composition techniques leave open the problem of constructing primitive morphisms.

Theories together with their morphisms define a category.

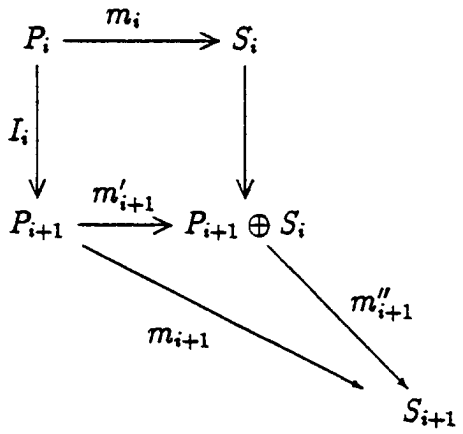
7.2 Refinement Hierarchy and the Ladder Construction

Abstract programming knowledge can be represented by theories. For example, we showed how to represent divide-and-conquer [33], global search (binary search, backtrack, branch-

and-bound) [34], and local search (hillclimbing) [22] as theories. The same approach can be applied to data structures [5], architectures [16], and graphical displays (e.g. Gantt charts).

A collection of problem-solving methods can be organized into a refinement hierarchy using theory morphisms as the refinement arrow [43]. See Figure 7.2. The question emerges of how to access and apply knowledge in such a hierarchy. The answer is illustrated in the "ladder construction" diagram in Figure 9.

The left-hand side of the ladder is a path in the refinement hierarchy of algorithm theories starting at the root (Problem Theory). $Spec_0$ is a given specification theory of a problem. The ladder is constructed a rung at a time from the top down. The initial arrow (theory morphism) from problem theory to $Spec_0$ is trivial. Subsequent rungs are constructed abstractly as follows:



where $P_{i+1} \oplus S_i$ is the pushout theory (shared union) and S_{i+1} is an extension of S_i determined by constructing the theory morphism m''_{i+1}. The morphism m_{i+1} is determined by composition.

7.3 Constructing Theory Morphisms

Constructing the pushout theory is straightforward. The main issue arising from this ladder construction is how to construct the theory morphism m''_{i+1} from the pushout theory to S_{i+1} (an extension of S_i). We formalized four basic methods for constructing theory morphisms last year, by analyzing the algorithm design tactics in KIDS [39]. Two of the techniques are well-known or obvious. However we identified two new general techniques for constructing theory morphisms: unskolemization and connections between theories. Roughly put, *unskolemization* works in the following way. A theory morphism from theory A to theory B is based on a signature morphism which is a map from the symbols of A to the symbols of B. A theory morphism is signature morphism in which the axioms of A translate to theorems of B. Suppose that during a design process we have somehow managed to construct a partial signature morphism – only some of the symbols of A have a translation as symbols of B. The question is how to derive a translation of the remaining symbols of A. The unskolemization technique uses the axioms of A and deductive inference to solve for appropriate translations

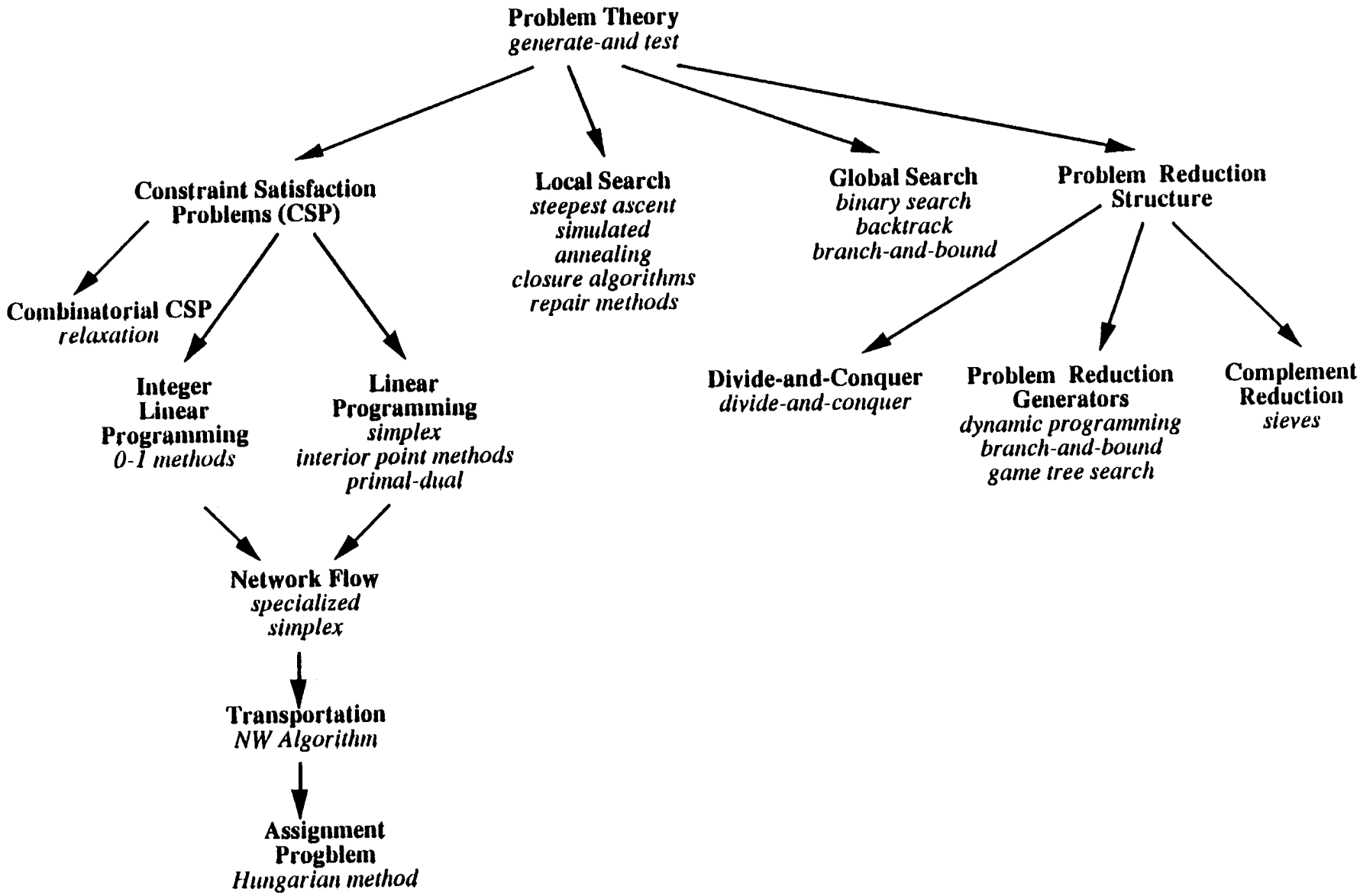


Figure 8: Refinement Hierarchy of Algorithms

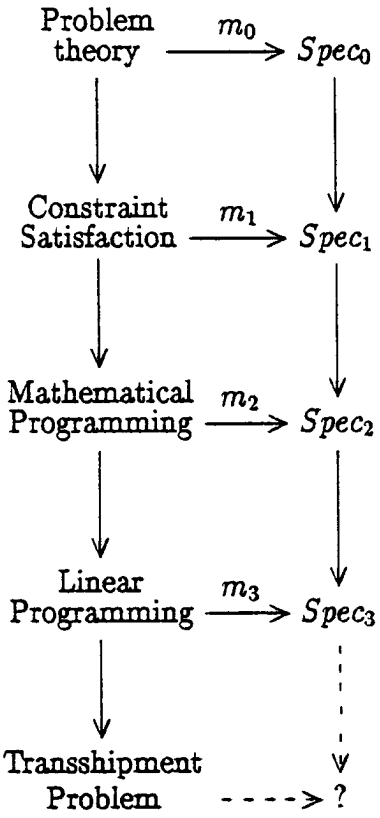


Figure 9: Classification Approach to Design

of these symbols. As a simple example, suppose that function symbol f is untranslated and that it is the only untranslated symbol in an axiom $\forall(x)G[f(x)]$ of A. We unskolemize f by replacing its occurrence(s) with a fresh existentially quantified variable: $\forall(x)\exists(z)G[z]$. This unskolemized axiom can now be translated and we can attempt to prove it in theory B. A proof yields a witness for the existential that is a term that depends on x. This term can serve as the translation of f knowing that such a translation preserves the theoremhood of the considered axiom. We may need to verify other axioms involving f to assure the appropriateness of the derived translation.

This technique underlies the problem reduction family of algorithms and tactics in KIDS. For example, in constructing a divide-and-conquer algorithm we need to find translations of decompose, solve, and compose operators. The tactic works by letting the user select a standard decomposition operator from a library (or dually, selecting a standard compose operator) and then using unskolemization on a "soundness axiom" that relates decompose and compose. The unskolemized soundness axiom can then be proved in the given problem theory to yield a specification of the compose (resp. decompose) operator. To be more specific, if we are deriving a divide-and-conquer algorithm for the sorting problem, then we want to construct a theory morphism from divide-and-conquer theory into sorting theory. We might choose a standard decompose operator for input sequences, say split-a-sequence-in-half, and the unskolemization technique leads to a derivation of a specification for the usual merge operation as the translation of compose. The result is a mergesort algorithm. Other choices leads to quicksort, selection sorts, and insertion sorts [33].

Sometimes the axioms of a theory are too complex to allow direct application of unskolemization. This situation arises in the theory of global and local search algorithms. We have discovered and developed recently the concept of *connection between theories* which underlies and generalizes our correct but somewhat ad-hoc solution to this problem in the global and local search design tactics. The general result regarding connections between theories is this: Suppose that there is a theory T from which we want to construct a theory morphism into a given application domain theory B. If there is a (preexisting) theory morphism from T to a library theory A and we can construct a connection from A to B, then we immediately have a theory morphism from T to B. So connections between theories are a way to adapt a library T-theory to a new, but related problem.

The concept of connection between theories relies on several ideas. The sorts of a theory are all interpreted as posets (including booleans) and furthermore the set of sorts itself is a poset (under the subsort partial order). A collection of "polarity rules" are used to express (anti-)monotonicity properties of the functions and predicates of a theory. For example, $size(\{x \mid \neg P\})$ is monotonic in $\{x \mid \neg P\}$ but antimonotonic in P; so if $Q \implies P$ then $size(\{x \mid \neg P\}) \leq size(\{x \mid \neg Q\})$. These polarity rules are used to analyze the axioms of a theory and then to set up various connection conditions between the corresponding operators of theories A and B – these conditions directly generalize the conditions of a homomorphism. Furthermore the polarity analysis is used to direct conversion maps between corresponding sorts of A and B. Given these conditions and conversion maps it can in general be shown that the axioms of A imply the corresponding axioms of B, thus establishing the theory morphism.

We have prototyped this classification approach and have tested it on some simple problems.

Steve Westfold built a graphical interface to the refinement hierarchy that allows graphical navigation of it and incremental application. Jim McDonald developed a simple Theory Interpretation Construction Interface that supports the development of views (theory interpretations or morphisms). It shows source and target theory presentations and the current (possibly partial) view between them. Users have several tools to support the completion of a view, including typing in translations for various source theory symbols and using "unskolemization" (one of the four basic methods mentioned above). We demonstrated the use of this system to develop a view from divide-and-conquer theory into a simple problem theory. A more complete implementation of these techniques is underway in the Specware system at Kestrel [50].

8 Concluding Remarks

Our original conception of the scheduling effort has evolved in significant ways. Our 1991 demonstration system was based on use of a general-purpose object base manager and the compilation of declarative constraints into object base demons. We also used a Simplex code to check feasibility of start-times in a generated schedule. The results were somewhat disappointing in that for the CDART problem we obtained from CMU, our first code couldn't solve it running overnight, and our second code could only solve most of it using several minutes time. The derived scheduler described in this paper finds a complete feasible solution to the same problem in less than one second.

Since speed is of the essence during the scheduling process and the object base and Simplex algorithm are problem-independent, it seemed wise to exploit our transformational techniques to try to derive codes that are problem-specific and highly efficient. Rather than compile constraints onto an active database, we now derive pruning mechanisms and constraint propagation code operating on problem-specific data structures. Rather than use a Simplex algorithm for finding feasible start-times, the constraint propagation code maintains feasible start-times throughout the scheduling process. The advantage of our approach is the ability to expose problem structure and exploit it by transformationally deriving efficient problem-specific code.

References

[1] ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM 26*, 11 (November 1983), 832–843.

[2] APPLEGATE, D., AND COOK, W. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing 3*, 2 (Spring 1991 1991), 149–156.

[3] ASTESIANO, E., AND WIRSING, M. An introduction to ASL. In *Program Specification and Transformation*, L. Meertens, Ed. North-Holland, Amsterdam, 1987, pp. 343–365.

[4] BIEFELD, E., AND COOPER, L. Operations mission planner. Tech. Rep. JPL 90-16, Jet Propulsion Laboratory, March 1990.

[5] BLAINE, L., AND GOLDBERG, A. DTRE – a semi-automatic transformation system. In *Constructing Programs from Specifications*, B. Möller, Ed. North-Holland, Amsterdam, 1991, pp. 165–204.

[6] BROY, M., WIRSING, M., AND PEPPER, P. On the algebraic definition of programming languages. *ACM Transactions on Programming Languages and Systems 9*, 1 (January 1987), 54–99.

[7] BURSTALL, R. M., AND GOGUEN, J. A. Putting theories together to make specifications. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (Cambridge, MA, August 22–25, 1977), IJCAI, pp. 1045–1058.

[8] CAI, J., AND PAIGE, R. Program derivation by fixed point computation. *Science of Computer Programming 11* (1989), 197–261.

[9] EHRIG, H., KREOWSKI, H. J., THATCHER, J., WAGNER, E., AND WRIGHT, J. Parameter passing in algebraic specification languages. In *Proceedings, Workshop on Program Specification* (Aarhus, Denmark, Aug. 1981), vol. 134, pp. 322–369.

[10] EHRIG, H., AND MAHR, B. *Fundamentals of Algebraic Specification, vol. 2: Module Specifications and Constraints.* Springer-Verlag, Berlin, 1990.

[11] ET AL., J. S. *A Review of Strategic Mobility Models and Analysis.* Rand Corporation, Santa Monica, CA, 1991.

[12] FOX, M. S., SADEH, N., AND BAYKAN, C. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (Detroit, MI, August 20–25, 1989), pp. 309–315.

[13] FOX, M. S., AND SMITH, S. F. ISIS – a knowledge-based system for factory scheduling. *Expert Systems 1*, 1 (July 1984), 25–49.

[14] GOGUEN, J. A., THATCHER, J. W., AND WAGNER, E. G. An initial algebra approach to the specification, correctness and implementation of abstract data types. In *Current Trends in Programming Methodology, Vol. 4: Data Structuring*, R. Yeh, Ed. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[15] GOGUEN, J. A., AND WINKLER, T. Introducing OBJ3. Tech. Rep. SRI-CSL-88-09, SRI International, Menlo Park, California, 1988.

[16] GRAVES, H. Lockheed environment for automatic programming. Tech. rep., Lockheed Palo Alto Research Center, Palo Alto, CA, 1990.

[17] GUTTAG, J. V., AND HORNING, J. J. The algebraic specification of abstract data types. *Acta Inf. 10* (1978), 27–52.

[18] HENTENRYCK, P. V. *Constraint Satisfaction in Logic Programming.* Massachusetts Institute of Technology, Cambridge, MA, 1989.

[19] HOARE, C. Varieties of programming languages. Tech. rep., Programming Research Group, University of Oxford, Oxford, UK, 1989.

[20] JÜLLIG, R., AND SRINIVAS, Y. V. Diagrams for software synthesis. Tech. Rep. KES.U.93.2, Kestrel Institute, March 1993. To appear in: *Proceedings of the 8th Knowledge-Based Software Engineering Conference*, Chicago, IL, September 20–23, 1993.

[21] LADKIN, P. Specification of time dependencies and synthesis of concurrent processes. In *9th International Conference on Software Engineering* (Monterey, CA, March 30–April 2, 1987), pp. 106–116. Technical Report KES.U.87.1, Kestrel Institute, March 1987.

[22] LOWRY, M. R. Algorithm synthesis through problem reformulation. In *Proceedings of the 1987 National Conference on Artificial Intelligence* (Seattle, WA, July 13–17, 1987).

[23] LOWRY, M. R. Automating the design of local search algorithms. In *Automating Software Design*, M. Lowry and R. McCartney, Eds. AAAI/MIT Press, Menlo Park, 1991, pp. 515–546.

[24] LUENBERGER, D. G. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[25] MINTON, S., JOHNSON, M., PHILIPS, A. B., AND LAIRD, P. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conferenceon Artificial Intelligence* (1990), pp. 290–295.

[26] NEMHAUSER, G. L., AND WOLSEY, L. A. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., New York, 1988.

[27] PARTSCH, H. *Specification and Transformation of Programs: A Formal Approach to Software Development.* Springer-Verlag, New York, 1990.

[28] SADEH, N. Look-ahead techniques for micro-opportunistic job shop scheduling. Tech. Rep. CMU-CS-91-102, Carenegie-Mellon University, March 1991.

[29] SANNELLA, D., AND TARLECKI, A. Extended ML: An institution-independent framework for formal program development. In *Category Theory and Computer Programming, LNCS 240* (1985), pp. 364–389.

[30] SANNELLA, D., AND TARLECKI, A. Toward formal development of programs from algebraic specifications: Implementations revisited. *Acta Informatica 25*, 3 (1988), 233–281.

[31] SELMAN, B., LEVESQUE, H., AND MITCHELL, D. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conferenceon Artificial Intelligence* (1992), pp. 440–446.

[32] SMITH, D. R. Data structure design. in preparation.

[33] SMITH, D. R. Top-down synthesis of divide-and-conquer algorithms. *Artificial Intelligence 27*, 1 (September 1985), 43–96. (Reprinted in *Readings in Artificial Intelligence and Software Engineering*, C. Rich and R. Waters, Eds., Los Altos, CA, Morgan Kaufmann, 1986.).

[34] SMITH, D. R. Structure and design of global search algorithms. Tech. Rep. KES.U.87.12, Kestrel Institute, November 1987.

[35] SMITH, D. R. KIDS: A knowledge-based software development system. In *Automating Software Design*, M. Lowry and R. McCartney, Eds. MIT Press, Menlo Park, 1991, pp. 483–514.

[36] SMITH, D. R. Structure and design of problem reduction generators. In *Constructing Programs from Specifications*, B. Möller, Ed. North-Holland, Amsterdam, 1991, pp. 91–124.

[37] SMITH, D. R. Transformational approach to scheduling. Tech. Rep. KES.U.92.2, Kestrel Institute, November 1992.

[38] SMITH, D. R. Classification approach to design. Tech. Rep. KES.U.93.4, Kestrel Institute, 1993.

[39] SMITH, D. R. Constructing specification morphisms. *Journal of Symbolic Computation, Special Issue on Automatic Programming 15*, 5-6 (May-June 1993), 571–606.

[40] SMITH, D. R. Toward the synthesis of constraint propagation algorithms. In *Logic Program Synthesis and Transformation (LOPSTR '93)*, Y. DeVille, Ed. Springer-Verlag, 1993, pp. 1–9.

[41] SMITH, D. R. Synthesis of high-performance transportation schedulers. Tech. Rep. KES.U.95.6, Kestrel Institute, March 1995.

[42] SMITH, D. R. KIDS – a semi-automatic program development system. *IEEE Transactions on Software Engineering Special Issue on Formal Methods in Software Engineering 16*, 9 (September 1990), 1024–1043.

[43] SMITH, D. R., AND LOWRY, M. R. Algorithm theories and design tactics. *Science of Computer Programming 14*, 2-3 (October 1990), 305–321.

[44] SMITH, D. R., AND PARRA, E. A. Transformational approach to transportation scheduling. In *Proceedings of the Eighth Knowledge-Based Software Engineering Conference* (Chicago, IL, September 1993), pp. 60–68.

[45] SMITH, D. R., AND PARRA, E. A. Transformational approach to transportation scheduling. In *ARPA/RL Knowledge-Based Planning and Scheduling Initiative: Workshop Proceedings* (Tucson, AZ, February 1994), pp. 205–216.

[46] SMITH, D. R., AND WESTFOLD, S. J. Synthesis of constraint algorithms. In *Principles and Practice of Constraint Programming*, V. Saraswat and P. V. Hentenryck, Eds. The MIT Press, Cambridge, MA, 1995.

[47] SMITH, S. F. The OPIS framework for modeling manufacturing systems. Tech. Rep. CMU-RI-TR-89-30, The Robotics Institute, Carenegie-Mellon University, December 1989.

[48] SMITH, S. F., FOX, M. S., AND OW, P. S. Constructing and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems. *AI Magazine 7*, 4 (Fall 1986), 45–61.

[49] SRINIVAS, Y. V. Algebraic specification for domains. In *Domain Analysis: Acquisition of Reusable Information for Software Construction*, R. Prieto-Diaz and G. Arango, Eds. IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 90–124.

[50] SRINIVAS, Y. V., AND JÜLLIG, R. Specware:^{tm} formal support for composing software. Tech. Rep. KES.U.94.5, Kestrel Institute, December 1994. To appear in Proceedings of the Conference on Mathematics of Program Construction, Kloster Irsee, Germany, July 1995.

[51] TURSKI, W. M., AND MAIBAUM, T. E. *The Specification of Computer Programs*. Addison-Wesley, Wokingham, England, 1987.

[52] VELOSO, P. A. Problem solving by interpretation of theories. In *Contemporary Mathematics*, vol. 69. American Mathematical Society, Providence, Rhode Island, 1988, pp. 241–250.

[53] ZWEBEN, M., DEALE, M., AND GARGAN, R. Anytime rescheduling. In *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control* (San Diego, CA, November 5–8, 1990), DARPA, pp. 215–219.

A Coordinating Resource-Constrained Planning in Large Organizations

Ted Linden
David Einav

linden@netcom.com
einav@good.stanford.edu

March 11, 1994

Abstract

This report develops four interrelated techniques for interactive planning in a large organization. It explores the use of market pricing mechanisms to help multiple planning cells allocate resources, resolve conflicts, and exploit synergies. The pricing mechanisms support distributed planning where human/machine planning subsystems cooperate to achieve common goals and there are many goals that may be contingent, uncertain, changing, and imperfectly-perceived. Trade-offs between goals are formalized in utility functions that are not not required to be additive but do have some additive structure in the form $\Sigma u(t)*mod(t,...)$ where u(t) is a function of how and when a single task is completed and mod(t,...) captures the non-additive effects of dependencies between tasks. This flexible structure makes it possible to represent many large organizational planning problems and to evaluate alternative plans as they evolve. The report explores the use of decision theoretic techniques to estimate the market price of resources when given a non-additive utility function. Problem solving with these non-additive utility functions focuses on heuristic search with statistical look-ahead techniques used to merge evidence from the utilities and constraints when making variable and value order decisions during the search.

Table of Contents

1. Introduction

Experience from real planning applications continues to push AI planning technology in new directions. This research focuses on four interrelated techniques for planning the actions of a large organization:

1. **Market pricing mechanisms** are explored as a way to handle the interactions and dependencies among separate planning cells that are cooperating to plan the organization's activities.

2. **Distributed planning and planning for contingencies** is a major feature of organizational planning. Market pricing mechanisms deal with many diverse goals that may be contingent, uncertain, changing, and imperfectly-perceived.

3. **Utility functions** that capture user preferences and represent soft constraints have some additive structure but are not fully additive. Utility functions in the form $\Sigma u(t)$*mod(t,...) allow natural representations of the users preferences and support incremental, heuristic problem solving methods.

4. **Statistical look-ahead** techniques that project resource contention are useful even with non-additive utility functions. The look-ahead is used to make variable and value ordering decisions during heuristic search.

These planning techniques directly address fundamental issues arising in practical applications like military crisis action planning. Movement toward these techniques began ten years ago with research to eliminate assumptions that had limited the applicability of classical AI planning technology. Research in the late 1980's showed how to deal with uncertainty by building plans that are reactive [Brooks 86], universal [Schoppers 87], contingent [Linden & Glicksman 87], and adaptive [Alderman 86,88]. This research extends previous research on planning under uncertainty by addressing additional issues that arise when planning the actions of a large organization. Specifically, this research addresses requirements arising from the following common characteristics of organizational planning problems:

- **Interactive and distributed.** Organizational planning is interactive with human planners in control *throughout* the planning process. It is usually not enough to involve humans only in problem setup and in solution review. The human users are often geographically distributed.

- **Resource allocation.** Organizational planning emphasizes effective use of available resources. Automated planning for these applications needs to exploit the special characteristics of *resource capacity constraints* and deal explicitly with the hard issues involved in allocating resources.

- **Optimization.** The automated planning must find a good plan, not just a plan. The logical-satisficing approach of traditional AI planning needs to be

augmented so optimization is *explicit* within the automated process and is not left entirely to the user.

These characteristics of organizational planning are leading to a shift away from purely logical-satisficing, qualitative planning to planning techniques more capable of dealing with uncertainty, resource allocation, scheduling, optimization, and other issues that have quantitative as well as qualitative characteristics.

1.1 Origins of Approach in Applications

Many of the planning techniques documented in this report generalize techniques that were applied successfully in real applications. The initial version of the non-additive utility functions combined with statistical look-ahead was developed for use Rome Laboratory's Advanced Planning System (APS). APS is now operational and is helping teams of Air Force personnel plan, coordinate, and schedule up to 2000 daily missions. The planning component of APS has an AI architecture, implements constraint propagation, and uses a non-additive utility function to guide its search. It was the military's first operational high level planning system to incorporate significant elements of AI planning technology.

Military crisis action and transportation planning require further generalization of these organizational planning techniques. Previous research on these generalizations is documented in earlier reports [Linden 91, Linden & Vrotney 92]. The interrelated set of techniques that have evolved have also been influenced by direct practical experience with other projects including projects to manage containers for a shipping company, schedule training missions for pilots, plan Army Corps-level maneuvers, support submarine commanders, and plan missions and maintenance activities for NASA. Other organizational planning problems that can use the same techniques include:

- Planning for manufacturing operations.
- Integrated planning in logistics, transportation, and distribution systems.
- Integrated planning for "just-in-time" operations.
- Complex project planning and scheduling for everything from construction projects to personnel scheduling.
- Disaster relief planning.
- Sensor planning and allocation.
- Automation of agents acting in a simulation.
- Job shop scheduling when there is a complex, multivariate utility function.

1.2 Technical Requirements

Experience with planning applications shows that the formalisms needed to represent and solve many problems require major extensions to those of classical planning. While some problems fit the classical planning mold; organizational planning problems typically have significant technical requirements in most if not all of the following areas:

1. **Uncertainty**. The planning situation is dynamic with ongoing changes in goals, preferences, and resources. Knowledge of current and projected states is partial and uncertain.

2. **Dynamic replanning.** The plan evolves over time as today's plan is executed and addition information enables further elaboration of plans for the future.

3. **Optimization and deciding what to do.** There are many goals and constraints (often hundreds or thousands) and not all of them can be accomplished with the available resources. Deciding what to do is part of the problem. With some difficulty, user preferences among the goals can be approximated by a utility function.

4. **Non-additive utility**. Except at higher levels of abstraction, the utility function is seldom completely additive. Some goals support or conflict with other goals, and there are hard and soft constraints between task assignments. Some sets of goals have a conjunctive all-or-none property, and some goals are alternatives with decreasing additional value once M out of N are accomplished.

5. **Imperfect models.** The formal problem model is imperfect and only approximates the real planning problem. As an individual problem is being solved, users discover instance-specific constraints that had not been foreseen. Ongoing imperfections in the formal model cannot be completely eliminated and are the fundamental reason why organizational planning must be interactive.

6. **Users in the planning loop**. Users do not want a black box planner that only allows them to control the problem setup and evaluate results. Domain experts have trouble formulating general constraints and prefer to focus first on finding a specific solution.

7. **Distributed.** Organizational planning draws on the experience of many experts who may be distributed geographically.

A goal of the current research is to extend existing planning techniques so that technical requirements in all these areas can be handled within a single planning formalism. The four planning techniques developed during this research extend the set of techniques available to the designers of practical applications.

1.3 Technical Background

There has been little or no domain-independent research that allows technical requirements in all the above areas to be handled within a single formalism. Much recent work on AI planning addresses technical requirements in the first and second areas. Operations research handles large optimization problems with the third requirement, but does not formalize methods for dealing with most of the other six technical requirements. Optimal solutions for large problems with non-additive utility functions are usually infeasible, and there has been relatively little work on obtaining "good" plans when the utility function is non-additive.

The fifth and sixth requirements are considerations for theoretical work because some problem solving techniques are better than others at accommodating and benefitting from user interaction. Planning methods that translate the problem into a form that is not intelligible to human planners do not benefit from human interaction during the problem solving. Relatively few organizational planning problems are well enough defined for the organization to be willing to turn the planning over to fully automated software.

To support technical requirements for distributed planning, it is important for automated processes to do more than automate individual planning cells, they must also help coordinate the activities of planning cells so locally generated plans will assemble into effective global plans.

Some practical planning applications have addressed technical requirements in all seven areas, but have not used a separable, domain-independent planning engine. For example, Rome Laboratory's APS addresses technical requirements in all of the areas listed above. (Requirements in the first and second area are handled in a replanning extension of APS that is not yet operational but does exist in prototype form.)

A decision theoretic approach to planning seems able to formalize all of the technical requirements, but there are both theoretical and practical problems in extending decision theory to handle not just one static decision but hundreds of interrelated decisions that evolve over time.

This report describes research on four interrelated techniques that extend other techniques already documented in the planning literature. Several of these techniques generalize ideas originally developed for APS. The four new techniques, when combined with existing planning techniques, address all the technical requirements listed previously. They also exploit simplifying features of organizational planning problems that are not exploited by classical planning.

1.4 Exploiting Simplifying Features of Organizational Planning Problems

Organizational planning often requires relatively little emphasis on dynamic creation of task plans (how to accomplish a goal) and more emphasis on resource

allocation (what can be accomplished) and scheduling (when it can be accomplished). For example, in many manufacturing applications such as job shop scheduling, the possible process plans (the sequences of generic steps needed to accomplish an individual job) are known at design time. The main problem is to instantiate generic process plans with resource assignments. In unusual circumstances, it may be useful to invent new process plans, but this is not the focus of most practical applications.

It is important to exploit the role that resources play in organizational planning. Many domain-independent planning systems have not exploited the following common features of resources:

- Some resources are available in the initial state and are consumed by the plan but are not produced within the time frame in which the plan will be executed.

- There are multiple instances of many resources and the instances are interchangeable.

When the problem can be formulated so many resources have either or both of these attributes, market-based resource pricing mechanisms become effective. A thesis explored in this report is that resource pricing mechanisms and statistical look-ahead techniques make it practical to solve complex organizational planning problems when many of the resources have these two properties.

Another common simplification in organizational planning problems is that the resources and other parameters assigned to tasks are constrained by the resource constraints and by a relatively small number of binary or other low order constraints. Once the resource constraints are dealt with, each task is independent of most other tasks—each task is involved in only a relatively small number of binary or other low order constraints with other tasks.

The planning techniques proposed here are most useful for problems that fall between those handled by traditional AI planning and those handled by operations research techniques alone. The techniques are useful when there are complex dependencies between tasks, many contingent tasks, and some dynamic creation of tasks—features that cause an explosion in the number of variables when formulated as an operations research problem. On the other hand, the structure of organizational plans are often relatively stable and most planning decisions involve allocating and scheduling resources for a relatively fixed set of tasks. Classical AI techniques are needed occasionally to improvise new solution structures, but much of the planning involves assigning a consistent and near optimal set of values (resources and times) to a relatively stable set of variables.

1.3 Key Technical Ideas

We assume that plans are generated concurrently by separate planning cells that coordinate their planning activities to achieve a good overall plan. A key goal is

to find simple coordination techniques so the separate planning cells can share resources, resolve conflicts, and exploit synergistic opportunities. Potential conflicts and sharing opportunities can be modeled by an appropriate kind of phantom resource, so interactions can all be modeled as resource consumptions or resource productions.

Techniques for distributed situation assessment, goal selection, and establishment of preferences between goals are outside the scope of this work. We assume there is a common measure of cost/utility and all planning cells know the utility of their goals and can communicate about utility in a common language. (They do not have to agree about the utility of each others goals.)

Planning cells receive resources to execute their plan to the extent that they can pay an appropriate price which is measured in terms of the utility they will achieve by using the resource. A cell manages each resource and sets the price at which it will be bought and sold. Each planning cell builds plans that maximize the utility it achieves after accounting for the resources it buys or sells. The goal is to let each planning cell generate local plans with confidence that its plan will merge successfully with plans from other cells.

Resource pricing mechanisms are compatible with many characteristics of organizational planning. Resource pricing is more effective for large scale problems with many goals and many instances of each resource. To handle the uncertainty involved of organizational planning, probabilities about the situation, the goals, and the utilities can be propagated by the pricing mechanisms. Contingency plans can be generated by local planning cells and included in the overall plan. Dynamic creation of resources, goals, and tasks is feasible.

Capturing the organization's preferences about goals and representing these preferences in a utility function are important issues. This research addressed only the representation side of these two issues. When representing utility, it is useful to distinguish:

- Preferences about the importance of goals,
- Binary (or other low order) constraints about how two (or more) tasks are accomplished, and
- Resource capacity constraints.

Many binary constraints are not absolute requirements but are preferences about what makes a better plan. We represent these soft binary constraints as part of a utility function. Since soft constraints merge into hard, absolute constraints, all binary and low order constraints are represented by a utility function with the form $\Sigma u(t)*mod(t,...)$ where $u(t)$ is a function of how and when a single task is completed and $mod(t,...)$ captures the non-additive effects of dependencies between tasks. The summation is over all the tasks to be performed. The

function u(t) captures the additive component of the utility of performing the task. The mod(t,...) component captures the effects of binary or other low order constraints about the relationship between t and other tasks. For example, if t is to be completed before t', then mod(t,t') is 1 if the plan satisfies this constraint and it is 0 if not. Soft constraints are represented by values between 0 and 1.

For resource pricing to be effective in practice, prices must converge rapidly toward an equilibrium. To accomplish this, we explored the idea that planning cells should bid for resources by giving a probability distribution about the price they will be willing to pay or receive. When there is uncertainty about what other resources will cost and about what the best local plan will be, the local planning cell has only probabilistic information about the price it will want to offer for this resource. A large amount of previous work on statistical look-ahead techniques for AI scheduling has explored bidding for resources in terms of subjective probabilities of use [Muscettola & Smith 87, APS 89, Sadeh & Fox 89, Sycara et al. 90, Sadeh 91, Sadeh and Fox 91, Johnston 92]. We propose bidding in terms of both the price to be paid and the probability of use at that price. Section 8 summarizes the current state of research on a theoretical framework for this approach using concepts from decision theoretic planning.

1.6 Summary of Report

Section 2 is a review of military crisis action planning and an explanation of why formulating and solving practical crisis action planning problems requires the new planning techniques. It includes examples of crisis action planning problems that motivate the techniques. Section 3 describes how the new technique relate to each other and how they extend existing techniques to create an overall approach for solving organization planning problems. Section 4, 5, 7, and 8 cover the individual planning techniques while Section 6 gives examples of planning problems that are and are not handled by simple market mechanisms. Section 4 summarizes existing microeconomic theory on market mechanisms, identifies the limitations of current theory when addressing general planning problems, and defines necessary extensions. Section 5 contains results showing how distributed planning using market mechanisms handles contingent goals. Section 6 uses an extension of a traditional blocks world problem to identify the limits of market mechanisms. Section 7 summarizes work to develop utility functions that represent user preferences in a natural way, are useful during heuristic search to evaluate partial solutions, and enable effective variable and value ordering heuristics. Section 8 summarizes the results on generalizing and formalizing the decision-theoretic approach to heuristic search when using non-additive utility functions.

2. Military Crisis Action Planning

Military crisis action planning is one of the application domains that motivated the combination of planning techniques proposed here. This section summarizes the key features of this domain and justifies the claims made in the introduction about the need for the proposed planning techniques.

2.1 Crisis Action Planning is Interactive and Distributed.

Crisis action planning needs to be interactive in a stronger sense that what is implemented in most "interactive planners." Interviews with crisis action planners indicate that they want to be in control so they can handle the unique characteristics of this crisis. They do not want to work around a system that is using methods inappropriate for the current situation. Too many interactive planners put users in the role of helping the system rather than the system in the role of helping users.

Most techniques from operations research allow user interaction only during problem setup and plan evaluation. The planning algorithm is a black box. When all the important aspects of the user's real problem are captured in the formal model, optimization techniques solve the user's problem. However, in crisis action planning, it is usually impossible to know in advance all the goals, constraints, and preferences that will become important as a specific problem is solved. Details that are normally irrelevant may become critical in specific problem instances. Interpreters for an obscure language, overflight rights, and backup power supplies can quickly become critical to plan success. During Desert Storm, it became important to have turkeys to serve on Thanksgiving. Experienced users are good at handling these details as they encounter them, but they find it impossible to identify all the details that may be relevant to all possible planning situations. User interaction needs to occur during plan development whenever the formal problem model remains imperfect. It is very inefficient for the user to wait until a plan is completed, notice that it violates an unforeseen constraint, and then go back to the problem setup and try to change the setup in a way that will produce an acceptable solution.

Automated planners for use in crisis action planning should be designed to enhance an existing partial plan incrementally. They should take a partial plan as input (along with goals, constraints, initial conditions, and utility functions) and transform it into an enhanced partial plan as output. Such a planner can be used in a fully interactive mode. Users can generate an initial partial plan and let the automated planners extend the partial plan. Different automated/human planners can work on different pieces of the plan—each focusing on their areas of expertise. In general, automated planners will solve subproblems as they become well defined. Fully automated planning is still feasible with this transformational approach—starting from the goals, the automated planner can be called recursively until a plan is complete; however, the important feature of this approach is that users control the outcome of the planning process by

building portions of the plan themselves rather than by manipulating setup information.

This incremental, transformational approach also conforms with the distributed nature of crisis action planning. Users at different locations each have their own areas of expertise and are most effective when dealing with certain types of subproblems. Automation should help users deal not only with their subproblems, but also with the harder problem of making their subplans be effective as part of the larger plan.

2.2 Crisis Action Planning is Mostly Resource Allocation.

Crisis action planning is heavily a resource allocation problem. If resources were unlimited, feasible plans to resolve the crisis would usually be clear. The hard problem is that resources are scarce and must be shared. Much of the coordination between distributed planning cells is either a resource conflict or can be modeled as one. For example, if an action planned by one cell has to occur before an action planned by another cell, this constraint can be modeled by a phantom resource that is produced by the first action and consumed by the second action

In crisis action planning, airlift capacity is one of the scarce resources to be allocated. Airlift capacity has the two simplifying properties of resources mentioned in Section 1.4:

1. The planners have very limited control over the number of aircraft available. New aircraft cannot be built during the time frame of the crisis. Repair activities and leasing of commercial aircraft provide a limited set of options for increasing the available aircraft.

2. Any of many suitable aircraft will do. (While outsize equipment can only go in C5As, there are still many instances of C5As.) The planners only care that they will get an appropriate quantity of airlift capacity at the appropriate time. For many planning purposes, it is useful to allocate a quantity of airlift capacity and relegate to lower level planning detail the question of what cargo will go on which actual aircraft.

To the extent that distributed crisis action planning cells agree on the relative importance of their goals, resource pricing is a relatively simple way for them to communicate and coordinate their planning. Getting agreement about the relative importance of goals is a problem as discussed in Section 2.3. In organizations like the military with a strong hierarchical structure, setting priorities for goals is one of the functions of the hierarchy.

2.3 Optimization Issues in Crisis Action Planning

Crisis action plans must be good in the sense that they accomplish as much as possible with the available resources. There will always be more things to do

than resources to do them, part of the planning problem is deciding on a coherent set of actions that are feasible with the available resources.

At the highest levels of abstraction, crisis action plans may involve only a few key goals. For example, an abstract plan might be to defend a sector from a frontal attack, protect both flanks, and then launch a counterattack from the southern sector. At this level of abstraction, there are only a few goals, however, there are hundreds of constraints that determine the quality and acceptability of the plan. Constraints may deal with the suitability of the forces for the specific mission and terrain; they may come from the readiness of the forces or from the transportation needed to move them into place. Additional preferences for force assignments may consider recent commitments of the forces and other subjective considerations like the effectiveness of their commanders in working together.

Automation that merely assigns resources to an abstract plan does not accomplish much that users cannot do themselves. Automation is more helpful at lower levels of abstraction. At lower levels of abstraction, there are likely to be hundred or thousands of goals to be accomplished. A U.S. military transportation plan currently starts out as an uninstantiated TPFDD with 10,000 to 200,000 items to be delivered. Each item to be delivered is a goal/task to be accomplished. While the transportation plan should be built with a hierarchical structure that collects related goals, effective automation of the planning has to deal with very large numbers of goals.

Crisis action planning involves large numbers of constraints. It is useful to distinguish resource capacity constraints from the constraints arising from conflicts or dependencies between pairs of tasks. Resource capacity constraints limit the assignments of a resource to be no more than the available instances of the resource. Resource capacity constraints create a potential interaction between all of the tasks that can use that resource. For example, an airport capacity constraint means that every decision to send an item through that airport has a potential impact on the ability to deliver every other item that might pass through that airport. On the other hand, constraints that arise from conflicts or dependencies between tasks are usually binary or low order constraints. For example, a requirement that surgical facilities have to arrive before the surgeons is a binary constraint between those two delivery requirements.

Most constraints are not absolute prohibitions. For most constraints, domain experts can think of a situation in which it would be appropriate to violate the constraint. These constraints are actually preferences with a strong penalty against overall utility if the constraint is violated. There is a fine line between hard constraints and preferences.

2.4 Uncertainty and Contingency Planning

Crisis action planning is complicated by a large amount of uncertainty about both the situation and about the goals. Many of the delivery requirements in a TPFDD are needed only to handle contingencies, and some are far more important that others. Crisis action planning deals with many contingencies. "Are we prepared to deal with an attack from the north?" "What if we are not given permission to overfly the neighboring country?" "What if the monsoon arrives early?" Crisis action plans deal with hundreds of these contingencies. Each contingency typically leads to additional requirements. Many of the items included in a TPFDD are there to handle contingent goals.

One way to reduce the airlift requirements when carrying out a crisis plan is to plan more carefully for contingencies. Rather than taking redundant resources into the theater of operations for different contingencies, it is often possible for a single resource to handle any one of several possible contingencies. This requires explicit planning for contingencies at a fairly low level of detail. However, the payoff is significant in that it can greatly reduce the airlift capacity needed to get the required forces into position. The proposed planning techniques deal with large numbers of contingent goals and exploit the opportunities to share resources among multiple contingent goals.

2.5 Crisis Action Planning Examples

Simple but realistic crisis action planning examples were developed as a way of experimenting with the new planning techniques. Transportation plans that instantiate TPFDDs were one source of examples. The Kestrel scheduler assigns and schedules resources for the delivery requirements of a TPFDD. One application of the new planning techniques will be a front end to the scheduler that builds a list of delivery requirements hierarchically and explicitly represents the constraints and dependencies between hierarchical delivery requirements.

Another set of examples focuses on planning for contingencies in ways that exploit the opportunity to handle multiple contingencies with a single set of resources. These examples deal with planning for operations in theater as well as transportation planning.

The scenario is that a neighboring country is threatening to attack a friendly country which has appealed for U.S./NATO assistance. This scenario is an abstraction of a real training scenario. It is deliberately left abstract to avoid making it sensitive in any way.

The U.S. has decided to send in forces sufficient to deter the threatened attack. The situation is shown in Figure 2-1. In this simplified example, there are two goals:

1. Defend against a threatened incursion in Sector 1. Enemy forces are already massed to attack in Sector 1.

2. Be prepared to defend against a possible incursion in Sector 2. Intelligence is confident that they can give 5 days warning before enemy forces could be repositioned to attack into Sector 2.

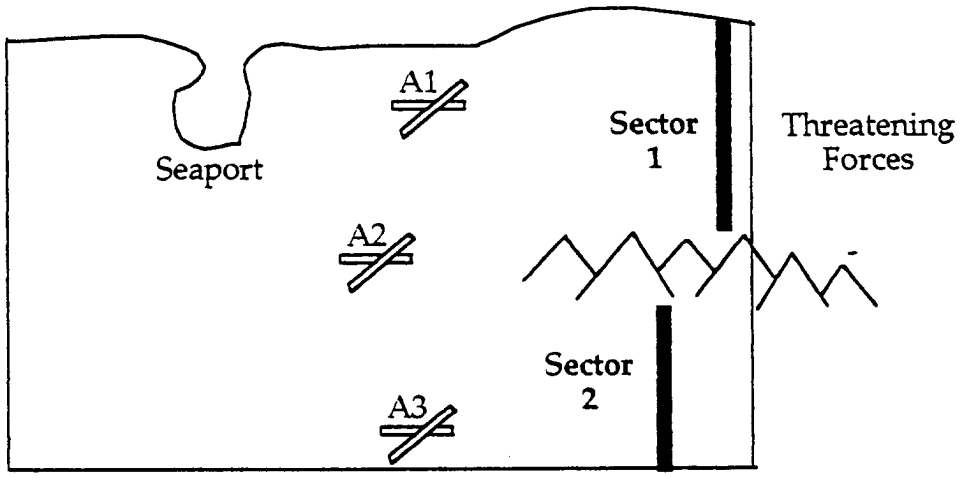


Figure 2-1: Crisis action planning scenario

Separate planning cells are dealing with the defense of each sector. Both planning cells develop a high level plan to send a force to secure the seaport and the perimeter around a supporting airport, and then send in the main air and ground forces to defend their sector. The interesting technical issue in this scenario is to enable the two planning cells to coordinate their plans to avoid conflicts and to exploit opportunities for synergy.

In this scenario, there is clearly synergy between the two plans in that they both need to secure the seaport, and this needs to be done only once. There are two other opportunities for synergy and three potential conflicts at this level of abstraction. (There may be more opportunities for synergy or conflict at lower levels of abstraction.) The opportunities for synergy are:

1. Since there will be 5 days warning before an attack in Sector 2, some forces from Sector 1 can be shifted to reinforce Sector 2 before an attack there. (Intelligence is confident that any threat in Sector 2 will significantly weaken the threat in Sector 1.)

2. One airport may be able to support the forces in both areas.

The potential conflicts at this level of abstraction are:

1. The transportation resources to bring forces into the theater of operations may not be able to support both plans simultaneously.

2. Simultaneous execution of both plans may cause congestion in the seaport and connecting roadways.

3. An airport supporting action in both sectors may become congested.

The goal of this planning exercise is to allow each planning cell to plan the operations for its sector with as much independence as possible while still identifying and exploiting the synergies and dealing with the conflicts between the separate plans.

Software was implemented to experiment with a variety of planning problems based on this overall scenario. This report focuses on how the problem solving works when the two planning cells choose airports to support their separate operations. Depending on the relative advantages of the different airports, the best plans may be for each operation secure the airport that is best for its own purposes; however, sometimes it is better to use a single airport and exploit the concept of a common good. The key issue is how two relatively independent planning operations can decide that they are each better off securing a single, common airport.

To be more specific, assume the following facts about the situation. The cost of securing each airport is the same. (The cost of securing an airport is measured in terms of the quantity of forces that must be sent in to secure the airport, the time required, and other intangible factors like the political implications of using the airport.) Airport A1 is slightly better for supporting operations in Sector 1, but Airport A2 is a viable alternative. (The utility of using an airport considers factors like the distance to the area of probable operations, the quality of the facilities at the airport, and similar domain specific issues.) Airport A3 is slightly better for supporting operations in Sector 2, but Airport A2 is also a viable alternative. If the planning cells act independently, the first one will develop a plan to secure the seaport and Airport A1 and then send forces to defend Sector 1. The second planning cell will develop a similar plan using Airport A3.

Both planning cells need to discover that securing the seaport is an operation common to both plans. It needs to be done only once, but congestion in the common seaport may also create conflicts between the two operations. This is accomplished in the software by representing the seaport as a shared resource.

The more interesting question is how the two planning cells discover that they are both better off using A2 because they can share the cost of securing A2. (By transporting fewer forces to secure airports, the main forces to defend the sectors will begin arriving sooner.) Each of the airports is treated as a shared resource, and there is a cost of securing and using that resource.

Coordination between the planning cells requires:

1. Identifying resources that are potentially shared.

2. Having each planning cell bid for the shared resources.

In this example, each planning cell will bid a lower amount toward the cost of securing A2; however, if the sum of the bids for A2 is higher than the individual bids for the other airports, A2 becomes the airport preferred by both planning

cells. The independent planning cells exchange information about shared resources in terms of quantitative utility measures. Each cell is trying to maximize the quality of its own subplan. The pricing of shared resources allow coordination between the planning cells so the individual plans developed in each cell combine into a good overall plan. In this case, each planning cell discovers that it needs to pay only part of the cost of securing A2, and thus A2 becomes its preferred airport. In slightly different circumstances where using A2 is distinctly inferior for one or both of the planning cells, if the sum of the bids for A2 are lower than the cost of securing A2, the planning cells will choose to secure both A1 and A3. If congestion from sharing A2 reduces its usefulness to one or both of the planning cells, the planning cells will reduce their bids for A2 which may cause A2 to become unattractive to each of the planning cells.

In this case, a market pricing mechanism leads independent planning cells to plans that combine effectively into a good overall plan. By identifying and pricing other potentially shared resources, the other synergies and conflicts in this scenario can also be planned successfully. Some of the main forces sent into the theater can be identified as potentially shared between the two sectors. (The forces will reinforce in Sector 1 until intelligence reports that the enemy is shifting forces toward Sector 2.) Pricing mechanisms allow the appropriate trade-offs to be made. For example, in planning the defense of Sector 2, the value of shared rescues would be greatly reduced if the intelligence warning is less timely or less certain.

The market pricing mechanisms implemented for this example are intended to scale up to larger problems where there are many planning cells and many potentially shared resources. Indeed, pricing mechanism actually work better on larger problems with many entities competing for a large pool of shared resources. However, there are limits to the effectiveness of market pricing mechanisms in the context of general planning problems, and this research explored those limits.

3. Overview of the Planning Techniques and their Interactions

The overall approach pursued in this research assumes that plans are generated by multiple, semi-independent planning cells. At one extreme, these planning cells may be geographically distributed teams of human planners. At the other extreme, the planning cells may be small software modules that make one specific kind of elaboration, change, or transformation to a plan. Planning cells may lie anywhere in the spectrum between these two extremes. They may use dynamic programming or other algorithms from operations research, they may be expert systems, they may be interactive planning cells, or they may be individual human planning experts. The key problem is to find simple coordination techniques so the separate human and automated planning cells can coordinate their planning activities, share resources, resolve conflicts, and exploit synergistic opportunities.

3.1 Dividing Problems into Semi-independent Subproblems

When a problem divides into completely independent subproblems, it is relatively easy to solve each subproblem and build a complete solution from the subproblems. Mathematical expression simplifiers and transformational compilers for very high level languages are the most sophisticated examples of exploiting this ability to decompose a problem into many independent subproblems. These applications are implemented as a set of transformations that pattern match on syntactic expressions of a formal language and then transform the expression into a simpler or more executable expression. These applications can be built so most transformations do not interact with other transformations.

Practical planning problems divide into only *semi*-independent subproblems. Resolving conflicts between plans that achieve conjunctive goals has been a central theme of planning research. This research explores the extent to which resource pricing mechanism allow semi-independent planning cells to coordinate their activities. It models the interactions between planning cells as resource conflicts or resource sharing opportunities and lets the planning cells communicate by bidding for alternative resources. Communication of resource price information may not be adequate to coordinate all distributed planning; however, it is interesting to see how much coordination can be accomplished with simple resource pricing mechanisms.

Organizational planning problems often divide into subproblems that are independent except for access to shared resources. Typically, the organization is already structured in a way that attempts to minimize interactions between planning cells. Expensive resources that must be shared are the most common cause of interactions. Pricing mechanisms using ideas from microeconomic theory are an established way to allocate resources efficiently. Section 4 explores the extent to which these ideas apply in organizational planning problems like military crisis action planning.

Dependencies between planned tasks that are not explicitly over resources can still be modeled as resource dependencies. For example, if there is a constraint that Task A has to be completed before Task B, this can be modeled by a phantom resource that is produced by A and consumed by B. All dependencies between subproblems can be modeled as resource constraints by inventing appropriate phantom resources. Thus, it is plausible that communication between planning cells can be limited to that which is needed to identify and price shared resources.

From a practical viewpoint, it is not necessarily a simplification to translate all dependencies into resource constraints. Phantom resources increase the number of resource types, and most phantom resources have only one instance that is both produced and consumed by planned actions. Pricing mechanisms are less likely to be effective for these resources. However, in many organizational planning problems, most of the dependencies between planning cells arise over access to a static set of resources that have multiple instances. Pricing mechanism are effective for these resources, and it seems useful to extend the pricing mechanisms to handle a small number of other kinds of dependencies all within a single framework.

3.2 Problem Solving Steps

Organizational planning involves three steps. Sometimes they will be sequential, but they may be interleaved:

1. The organizations goals are apportioned to local planning cells. High level goals may be defined in the charter of the planning cell and detailed goals may be apportioned dynamically. From a practical viewpoint, planning is simplified to the extent that this apportionment minimizes the interactions between the planning cells.

2. Identify the potential interactions between the plans of separate planning cells. These interactions take the form of shared resources, conflicts, or synergies. Potential conflicts and synergies can be modeled by phantom resources. Many potential interactions (and their associated resources) are known at design time, other interactions have to be identified dynamically.

3. Let each planning cell develop a plan to solve its local goals in a way that will combine with other plans to become an effective plan for the entire organization.

The pricing mechanisms covered in this report deal with the hard part of the third step—the communications between planning cells so their separate plans combine effectively. The pricing mechanisms assume that all planning cells communicate about prices in a common language. Prices need not be in monetary units, but each planning cell must have an estimate of the utility of its goals—although the utility can be contingent, changing, and uncertain.

4. Pricing Mechanisms for Coordinating Planning Cells

This section introduces resource pricing mechanisms, reviews assumptions behind the efficiency theorem from microeconomic theory, and discusses the extent to which these assumptions are valid both in the economic world and when applied to organizational planning problems.

4.1 Resource Pricing Mechanisms

Resources are allocated to planning cells that are willing to pay an appropriate price which is measured in terms of the utility they will achieve by using the resource. Some authority is responsible for managing the price of each resource. This price control may be distributed with different authorities for separate resources. The authority assigns an estimated price for the resource for each day or other period of time. Planning cells then plan to accomplish their goals using resources that maximize the difference between the utility they achieve and the price of the resources. Each cell specifies the resources that it wants. When the resource authority receives the request, it calculates whether available capacity is over or under subscribed at the current prices, and it raises or lowers the price accordingly. Logically, each planning cell then recalculates its demands and the process cycles—hopefully toward an equilibrium point that allocates all the resources efficiently.

In large problems where demands for resources are reasonably independent and individual demands have an insignificant effect on prices, prices remain relatively stable unless affected by external factors. Price convergence for smaller planning problems is more problematic—especially when there are many conjunctive and disjunctive requirements for resources. For example, a task that needs either of two different packages of resources may switch between the resource packages when the price of one resource changes. If the problem is small, the change may have a ripple effect on the price of other resource.

When there are humans in the planning loop, multiple iterations on the resource prices are not feasible, and a key practical question is whether the initial price estimates with only a few modifications are good enough. When the planning is being done within an automated cycle, it is feasible to adjust the prices repeatedly in an attempt to find an equilibrium where the resource allocation is efficient.

4.2 Pricing Assumptions in Economics.

Microeconomic theory holds that the pricing mechanisms underlying the modern western economies leads to an efficient allocation of resources. This section fleshes out the assumptions behind the argument that these "free markets" are good. What is meant by "efficient allocation"? What basic conditions need be met to attain an ideal competitive market? What are the areas that do not fit the requirements of perfect markets? And it examines how

closely organizational planning problems can be modeled as problems in efficient resource allocation.

Formally, an efficient, or Pareto Optimal, allocation of resources is any allocation that uses up all the available resources in such a way that it is not possible to reallocate resources to make somebody better off without simultaneously making somebody else worth off. The definition does not mention the just or fair distribution, nor does it pay attention to general common goals. From the microeconomics perspective an economy is merely a collection of individuals each trying to maximize its profit. The multitude of individualistic actors leads, under some fairly general assumptions, to a globally efficient resource allocation.

There are 5 central assumptions behind the optimality of pricing mechanisms. They are:

- **Isolated utility.** Each participant in the global economy is basing his actions only on his "basket of goods." He does not care about other participants nor about the aggregate performance of the economy. For example, if somebody buys a foreign car it is because he perceives it to be the best value. A person cannot be expected to buy (or do) something only because it is good for somebody else, nor abstain from an activity (like pollution) because it makes others less well off.

- **Insaturation of demand.** Different participants may put different priorities on different goods. However, regardless of personal tastes and preferences, one thing is common to all. All participants feel they would be better off if they had more. We all want more.

- **Perfect information.** All participants have perfect information about the current market conditions.

- **Diminishing return to scale.** The return on investment in any production activity is a convex function that diminishes with the size of investment. For example, if we seed crops twice a year instead of once we will harvest less than twice the crops.

- **Insignificance of any individual player.** It is assumed that any participant cannot be significant enough compared to the market overall as to affect the price of the goods in the market. Any producer or consumer may not like the price they are getting or paying for the goods they sell or buy, but their decisions to buy or not are not significant enough to affect the price.

4.3 Economic Reality

The "free economy" model is widely hailed as being the best even though many of the assumptions above are unrealistic. Without going into a deep discussion on "managed competition," it is instructive to understand where the model brakes down in modern economic reality.

The first two assumption may be pretty close to reality. This is not to say that our charitable activities are not important, but rather that the Mother Teresas of our world, with all the great humanitarian work they do, have only marginal impact on the economical lives of most people.

The perfect information requirement is unattainable. The market economy is so vast and individual's resources are so minuscule in comparison that any attempt to collect all the information is doomed. However, one does not need all the information, but only some small portion that is directly relevant to its day to day decision making. On balance, one can realistically be expected to be adequately acquainted with the market conditions in his immediate area of trade.

The requirement of diminishing returns to scale is probably the weakest one in modern economy. It is clearly more efficient to construct large manufacturing facilities for goods with a significant component of R&D costs. The semiconductor and software industries are clear examples where the more one produces the less are the marginal production costs (this phenomenon is known as "learning by doing"). On the other hand, transportation and logistics problems generally do meet assumption.

The last requirement of insignificance of each player breaks down most readily when there is increasing return to scale but not only then. For example, recent economic history knew attempts to corner the world silver market.

4.4 Organizational Planning and Assumptions for Price Convergence

Like real economic systems, organizational planning problems do not meet the assumptions needed by microeconomic theory to guarantee that pricing mechanism will lead to an optimal allocation of resources. The first three assumptions—isolated utility, insaturation of demand, and perfect information are probably more valid for distributed, organizational planning problems than in the economy as a whole. However, the assumption that production is a diminishing, convex function of investment size is not valid for planning problems that involve discrete reasoning. More relevantly, resources seldom exist in sufficient quantities for the actions of individual planners to have an insignificant affect on prices. Certainly, when there is only one instance of a resources—as is the case with phantom resources—each planning cell can manipulate the price of the resource.

4.5 Summary on Microeconomic Theory

Economic reality is in many cases quite different from the assumptions required for perfect competitive markets. Optimal allocation of resources, if at all attainable, may have little to do with the norms of our society and the goal of enhancing the well being of all the members, and yet the free market model is widely popular and promoted as being the only solution for sustained growth of the modern economy. What is the indispensable feature that makes the free

market model so attractive, and makes countries that try to avoid it (like former Soviet Union, Cuba and India) pay the price of delayed economic development. The answer is in the works of the mechanism itself: it does not fail because it does not have any single point of failure. The decision mechanism is distributed and involves no single point of failure. The failure of any individual player has no significant impact on the economy overall. In comparison with the central command-type structure that existed in Soviet Union, the unattainable task of collecting all the information needed for successful decision making is avoided. The role of central authority is reduced to priority setting and coordination, while the bulk of detailed data is collected, evaluated and used for decision making in a distributed, local, and timely manner. This explains why the preceding discussion of economics has immediate and clear bearing on military crisis action planning.

In the next section we outline the components of a distributed crisis action planning architecture and suggest steps that might be taken to validate or disprove this proposal in the domain of the military crisis action logistic planning.

5. Distributed Planning for Multiple, Contingent Goals

Dividing and distributing the organizational planning problem is very common among successful large corporations. Distributed planning has similar advantages for military crisis action planning:

1. **Geography of large operations.** Like international corporations, military crisis planning spans continents.

2. **Uncertainties in dynamic planning.** Situations change at the pace that makes a swift local response indispensable. There is not enough time for central authority to deal with all the nuances and uncertainties involved in the local situations.

3. **Local specialization.** No matter how many satellite links are established, the local commander will have a better and deeper understanding of the local situation.

If the crisis action planning has much common with planning in modern competitive markets, the methods that have been successful in the later should be tested in the former. The following takes for granted the need for some sort of distributed decision making in crisis action situations. It focuses on a more technical question: what is needed to make the distributed planning work for crisis action planning.

5.1 Components for Successful Decomposition

In order to design a successful distributed management environment two elements must be addressed:

1. **Resource pricing.** Resource pricing of transportation resources can handle coordination between separate planning cells. The transportation task will be priced in much the same way as in commercial transportation except there will be more emphasis on the opportunity cost when using transportation resources. During a crisis, opportunity costs are usually more significant than operating costs.

2. **Local planning to use resources.** For a global distributed scheme to work, each local planner must be able to construct a plan that maximizes its local utility. Under conditions of uncertainty, the optimal plan is conditional on uncertain events, just as any corporate plan has various options depending on the success of development, market penetration etc.

We have developed a conditional planning algorithm that copes with the difficult task of conditional planning under uncertainty. This algorithm is general and may in principle be used in a centralized environment, but the computational complexity of conditional planning is very high, and it is important to do conditional planning within a limited scope. In this way,

decomposition and conditional planning compliment each other: the coordination scheme assures that each planning cell is facing only local and relatively small problems and is capable of creating an conditional plan, while the optimality of local plans enables global optimality through coordinated distributed planning.

5.2 Optimal problem decomposition

There are a number of ways to divide a large resource allocation problem into a set of smaller ones. Two principal ones are:

1. Price-directed decomposition.

2. Resource-directed decomposition.

In both cases the central authority functions as a market clearinghouse. The two methods differ in the nature of communication between the clearinghouse and local planners. In the first case the clearinghouse is responsible for setting a price for each resource. The local planning cells plan in the environment defined by these prices and tell the clearinghouse what and how many resources they want to buy or sell given the current set of prices. If the market is not in equilibrium (i.e. demand does not equal supply) the clearinghouse modifies the prices and the process is repeated until an equilibrium is reached.

With the resource-directed decomposition, a central authority communicates the resource allocation to each planning cell which is responsible to make an optimal local plan given its resource allocation, and to tell the clearinghouse the price it is willing to pay for an additional allocation of resource (or willing to accept to receive fewer resources). The clearinghouse modifies the allocation of resources and the process continues until the marginal prices for each resource reach equilibrium.

Both methods lead to an optimal global solution, and the choice of one or the other should depend on the implementation aspects of the problem. It involves deep understanding of the human factors in crisis action planning and cannot be made at the current stage of the project.

5.3 Optimal conditional plans

While local expertise and decision making are critical for in distributed operations, local efforts can still be assisted and automated. Each planning cell can use whatever planning algorithms are most suited to the local problem. There is no requirement in our approach to distributed planning that all the local planning cells use the same planning algorithms. Because the local problems are generally smaller in scope, planning methods that do not scale well may still be feasible. Conditional planning by stochastic dynamic programming is a particular method we use in the following example.

5.4 Utility-based conditional planning

In the previous sections we presented an outline of the distributed planning environment that is based on two elements: optimal resource coordination and optimal local planning. This section presents an example of optimal local conditional planning under uncertainty with resource constraints.

Our flexible planning methodology is based on the following assumptions:

- The goal of the planner is to maximize the plan's expected utility
- The utility of each proposed plan can be computed
- The solution space of the problem is known and represented by AND/OR graph
- The (possibly uncertain) costs and resource consumption of all actions in the graph are known
- All uncertainties pertinent to planning and execution actions are explicitly represented by appropriate probability distributions

Since multiple decomposition structures can be merged by introducing a higher-level OR-type node, the solution space of a problem is represented by a single AND/OR graph.

We do not distinguish the actions from the links that represent them. The semantics for the AND/OR graph are that all successors of an AND-type node must be satisfied to achieve it, while it is enough to satisfy any successor of an OR-type node.

Flexible planning minimizes the expected cost of the solution by finding an optimal exploration sequence for a given AND/OR graph. At each step a link is ``explored" in the graph, that is to say, the action it represents is carried out. We abstain from using the term "execution" in order not to imply that physical execution is taking place; a link may represent either a planning action that does not alter the physical environment, or some effector action, such as moving an object or activating a sensor.

The planner creates a resource-bounded strategy. While in this report we focus on time bounds, time is just one example of a finite resource. The results are valid for any finite number of resources such as time, fuel, manpower, or money, and if not otherwise stated, the reader can substitute `consumption of limited resource' for `time duration'.

Our current flexible planning methodology does not permit asynchronous interrupts. The basic model described here assumes that the actions cannot be interrupted at all. Readers interested in the treatment of arbitrary synchronous interrupts are referred to [Einav 91]. However, even the model we describe does

allow for one type of synchronous interrupt. As soon as an action violates a time constraint, it is interrupted, and an alternative solution is executed. We conservatively assume that the cost of the interrupted action is incurred up front.

Links are sequentially invoked by a solution process. A link may be invoked only once, and it may be selected only after one of its antecedents has been invoked. For a given set of previously explored links **h**, the set of links that may be invoked at the next step is called an active set and is denoted by act(**h**). The state of the solution process is defined by **h** and the remaining time **t**. The external time limit is denoted by T, and initially the history is empty. Obviously, not all link subsets correspond to possible histories, and/only those need to be considered when computing an optimal strategy.

A strategy is simply a rule assigning for each state a link to be explored at the next step. Our problem is to find for a given AND/OR graph the best strategy to compute and execute a solution within a given time.

An optimal strategy selects the next move using the potential value of the states that may result from that move. Thus an optimal strategy is intricately connected to the potential values in all possible states and is computed simultaneously with the table of potential values. To carry out this computation, we invoke the Bellman optimality principle [Bellman57] and stochastic dynamic programming.

The potential value table and the optimal strategy are computed gradually, starting with the potential values for one-move strategies, computing the two-move values on the basis of one-move values and continuing the recursion until the maximal-length optimal strategy is found. We will not present here the recursive equations. Instead we will present an example that will clarify the types of problems that we solve.

Consider the problem represented by the AND/OR tree depicted in Figure 5-1.

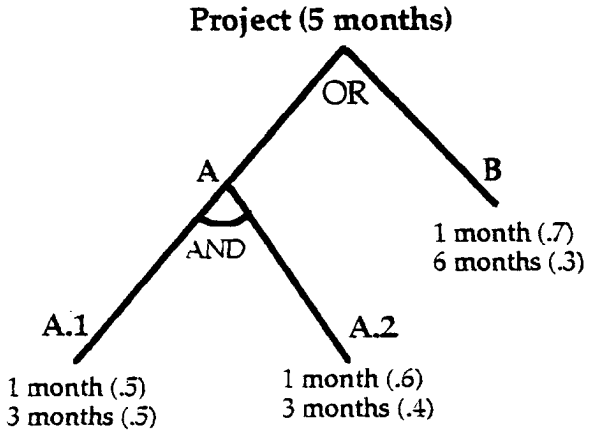


Figure 5-1: An AND/OR tree representing a simple project management problem.

Suppose that an experienced programmer is considering ways to approach a software development project that *must* be completed in five months. She recognizes two alternatives: A and B.

Alternative A splits the project into two modules: A.1 and A.2, while alternative B utilizes a new object class that is still being developed. Based on her experience with similar projects, the programmer estimates that it will take one or three months, equally probably, to encode the module A.1; the estimate is one or three months with probabilities 0.6 and 0.4 for module A.2. If alternative B is selected, the estimated time is one or six months with probabilities 0.7 and 0.3.

The focus on the control and selection of an appropriate solution is important when time bounds and uncertainty are present. In this situation we must ensure the feasibility of a solution, and the time uncertainty requires us to retain some degree of flexibility. Our problem is to find an optimal flexible strategy for the project.

The complete solution is presented in (Einav, 91). Here we only mention that although committing to one of the strategies leads to probabilities of 0.8 and 0.7 for timely completion of the project, the flexible optimal strategy has a probability of 0.88 of success.

Based on the assumptions outlined in the beginning of the section we designed a planner that can:

- Compute and execute an optimal meta-level control
- Manage optimally alternative decompositions
- Control optimally the uncertainty
- Plan optimally information gathering actions
- Interleave optimally planning and execution
- Find the resource-constrained conditional plan with the highest utility

The next step is to implement the flexible planner. The planner will be able to ensure that a system will produce relevant output in appropriate time; will represent explicitly and reason about temporal processes, including the problem solving process itself; will interact with the environment; and will adapt the reasoning process based on available resources.

The time-bounded optimal performance of the conditional planning methodology is achieved by controlling the concurrent exploration of the alternative decomposition structures.

Flexible control interleaves actions at different hierarchical levels. The optimal meta-level control is efficiently computed using the Bellman optimality principle within the framework of stochastic dynamic programming .

Flexible conditional planning aims to advance the state of the art in planning by addressing three major aspects of real-life applications: time constraints, uncertainty, and hierarchical structure of alternative decompositions and their abstractions.

Time constraints are always present in some form, either explicit or implicit, in real-life problems. We need to be able to react quickly while, time permitting being able to utilize slower, but more precise methods. As in design, where the duration and cost of the project are defined early in the life cycle, to control the time of reasoning we must start at the very early stages, *i.e.*, beginning with the selection of the appropriate hierarchical decomposition structures, and as long as we cannot predict exactly the performance of alternative decompositions and the environment, we have to model uncertainty. Our work led to a planner that differs in its basic assumptions from previous work in AI Planning.

6. Examples: Extending Pricing to Small Planning Problems

Most AI planning techniques work best on small problems and encounter difficulties of scale in larger problems. Resource pricing works best on a very large scale and encounters difficulties on smaller problems. This section gives examples that explore resource pricing in small problems where the resource requests of a single planning cell influences resource prices.

6.1 Job Shop Scheduling Experiment

One early experiment used the job shop allocation and scheduling problem discussed in [Sadeh 91b]. In involves 10 tasks to accomplish 4 jobs using 4 resources over 15 units of time. The tasks require resources for varying lengths of time. The solution requires careful use of critical resources with respect to the variable length time requirements.

We tried to solve this problem using simple pricing mechanisms and were surprisingly successful. We assumed a separate planning cell was planning each of the four jobs. We started with uniform prices for each of the 4 resources over the 15 time units. Each planning cell computed the cheapest way to accomplish its tasks within the allotted time. The times at which it choose to use resources created one unit of demand for the resources at those times. When a resource/time was demanded by more than one job, its price was raised. When it was not demanded by any job, the priced was lowered. Each planning cell then recomputed its cheapest solution using the new prices. Equilibrium prices were reached in 15-20 iterations. The union of the plans from each of the four planning cells was then the solution.

For this particular problem, iteration on resource prices seemed simpler than the statistical look-ahead technique used by Sadeh and Fox. The iteration was actually carried out manually for this problem.

6.2 Planning Problems without Equilibrium Prices

For small planning problems, there is no guarantee that equilibrium prices exist. The following is a simple example of a pathological case. There are four resources, R1-R4. Tasks A and B both need to use either of two pairs of resources as indicated in Figure 6-1. Task A will achieve 10 units of utility by using both R1 and R2, or it will achieve 2 units of utility by using R3 and R4. Task B will achieve 8 units of utility using either R1 and R3 or R2 and R4. There is no way to satisfy the resource requirements of both tasks, so the best solution is to let Task A achieve 10 units of utility. However, resource pricing mechanisms will result in low prices for R3 and R4 with the result that Task B can bid up the price for R1 and R2 to the point where Task A cannot afford both resources.

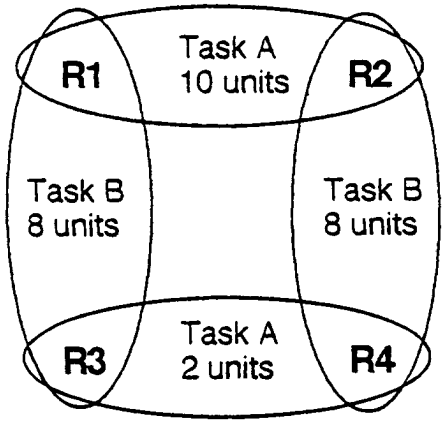


Figure 6-1: A planning problem without effective resource prices

6.3 Example: Contingency Planning in Blocks World

This section gives examples that show how market mechanisms apply to the problems traditionally addressed by AI planning technology. The market mechanisms deal with problems that cannot even be formulated in the terminology of classical planning, but it is important to understand the extent to which the new paradigm extends the older paradigms.

Blocks world problems can be formulated as resource allocation problems. When this is done, blocks problems have neither of the simplifying features that are exploited by resource pricing mechanisms (the resources are both produced and consumed, and there are seldom multiple instances of resources), thus there is little reason to expect pricing mechanisms to outperform old paradigms for these problems; nonetheless, it is informative to understand how pricing mechanisms apply to traditional problems.

6.3.1 Crisis Planning Analog of Blocks Example

Since a discussion of a blocks world problem has some danger of being viewed as irrelevant, it seems best to begin with an equivalent crisis action planning problem. A problem of moving forces into good defensive positions is illustrated in Figure 6-2. An enemy attack is expected toward Area 1, but there is a possibility that it will come toward Area 2 instead. Current defensive forces are an army battalion and an air wing in Area 3. Fuel has just arrived in Area 2. The goal is to get the battalion, the fuel, and the forces supporting the air wing to Area 1. They must move into Area 1 in that order. The fuel that just arrived in Area 2 is exposed if the attack comes into Area 2, so it should be moved out of Area 2 as soon as possible. An alternative but less effective goal is to defend in Area 3 and simply move the fuel to Area 3. The transportation resources are limited so only one thing can be moved at a time.

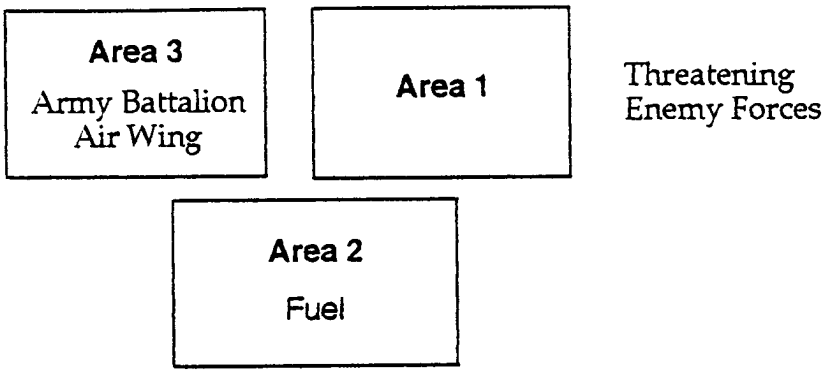


Figure 6-2: Crisis action planning scenario equivalent to blocks problem

There are three promising solutions to this problem:

1. Move the fuel from Area 2 to Area 3, then move the battalion, fuel, and air support forces to Area 1 in that order. This involves 4 movements and may take too long.

2. Move the battalion to Area 1, then the fuel from Area 2 to Area 1, then the air support forces. This involves only 3 movements but leaves the fuel exposed to danger while the battalion is being moved.

3. Move the fuel to Area 2 and satisfy the alternative goal with only one move.

The best choice depends on how soon the forces are needed in Area 1, whether the fuel really has to be moved out of Area 2 immediately, and the relative merit of the alternative goal of defending from Area 3

6.3.2 Blocks Example with Contingent Goal

This crisis action planning problem is equivalent to the following blocks world extension of the Sussman anomaly. Figure 6-3 shows the initial configuration of blocks. The three goals are to quickly stack A on B and B on C, and to put E on D within one move after the burglar alarm goes off. In parallel with the crisis action problem above, there are three promising solutions, and the best solution depends on the relative utility of the goals:

1. Move B to table, move E to D if the burglar alarm goes off, otherwise: move C to table, move B to C, move A to B. This requires 4 moves even when the burglar alarm doesn't sound and violates the quickly requirement for A on B and B on C.

2. Move C to table, move B to C, move E to D if the burglar alarm has gone off, move A to B. This means that E cannot be moved to D within one move if the burglar alarm goes off during the first step, but it satisfies all other goals.

3. Move B to C, move E to D if the burglar alarm goes off. This gives up the goal of A on B, but achieves the other two goals very quickly.

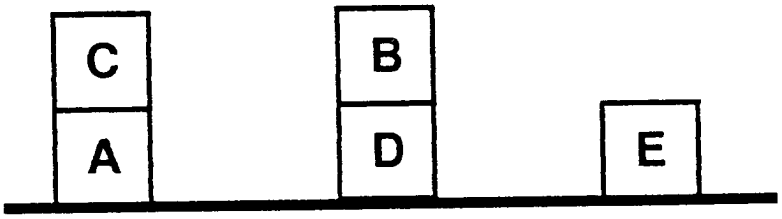


Figure 6-3: Initial arrangement of blocks

To choose the right plan, one needs to formalize the relative utilities of the three goals and do this as a function of the time the goal is accomplished. Knowledge of the probability that the burglar alarm will sound is also required. Formulation of utility functions for this problem is relatively straightforward. For example, the utility of each of the three goals may be given by the entries in the following table. Here A on B and B on C both have soft deadlines at the third turn, and E on D has a hard deadline in that it must occur on the first turn after the alarm in order to obtain any utility. The probability of the alarm sounding is .05 per turn. The utility of a plan is the sum of the utilities of the goals accomplished. The plan is to maximize the expected utility.

	Number of moves before goal is accomplished			
Goal	1	2	3	4
A on B	10	10	10	9
B on C	5	5	5	4
	Number of moves after alarm before goal is accomplished			
Goal	1	2	3	4
E on D	40	0	0	0

Table 6-1: Utility of each goal as a function of when it is accomplished

Changes in the utility of each goal and in the probability of the alarm sounding can lead to different choices for the best plan. While the utility function seems to make this problem more complex than traditional blocks world problems, in fact the utility function can help guide a problem solver toward a good solution.

6.3.3 Resource Pricing to Solve Blocks Example with Contingency Plan

To use market pricing mechanism to solve this problem, one can formalize it as a resource allocation problem with three agents competing for the resources. One agent deals with each goal. First, each agent develops one or more plans to accomplish its goal independently. In this case, these plans are quite simple (The notation Move(C,?) means that the agent doesn't care where the block is moved to):

Agent 1 plans to accomplish A **on** B by the two steps: **Move(C, ?), Move(A, B).**

Agent 2 plans to accomplish B **on** C by: **Move(B, C).**

Agent 3 plans to accomplish E **on** D by the two steps: **Move(B, ?), If alarm sounds Move(E,D)**

To accomplish these moves, each agent needs the resources of the robot arm at one or more times. Treat the robot arm as a resource for each of four units of time. The three independently generated plans involve potential conflicts and synergy. The conflicts are:

Move(C,?) has to occur before **Move(B,C)** if A **on** B is to be achieved.

Move(B,C) has to occur before **Move(A,B)** if B **on** C is to be achieved..

The potential synergy is:

Move(B,C) accomplishes **Move(B,?)**; however, moving B to the table accomplishes only the latter and not the former.

To allow the separate planning cells to coordinate their plans so they obey these constraints, one creates phantom resources that the cells can either supply or consume. We achieved the best results be creating phantom resources associated only with the time **Move(B,C)** occurs. All of the constraints happen to involve this operation. A planning cell can either get credit for supplying the operation **Move(B,C)** at a time desired by other planning cells, or it can offer to pay another cell to accomplish **Move(B,C)** at a specific time. There are eight resources that can be bought and sold by the three planning cells: the use of the robot arm at each of four time intervals, the specific movement of B to C at each of 4 possible time intervals. Note that one planning cell can choose to buy the robot arm to move B to C and simultaneously sell to other planning cells the accomplishment of **Move(B,C)**.

6.3.4 Results and Lessons Learned

We discuss lessons learned from several experiments with this simple resource pricing approach to blocks world planning. The research project was terminated before general conclusions were reached.

We began with arbitrary starting prices (often a uniform price of 2 units for each of eight resources) and had each planning cell choose to instantiate its plan in the way that maximizes its utility. When there were more consumers than suppliers of a resource, its price was raised (by .3 units in most experiments). When there were more suppliers, the price was lowered by a similar amount. This process iterated until prices stabilized or cycled.

With the eight resources described above, the planning cells arrived at the optimal plan relatively quickly. By changing the utility matrix, the plans converged to each of the three different likely solutions—even when one solution was only 1 unit better than another. Convergence took longer when two plans were almost equally good or when starting prices were far away from equilibrium prices.

In initial experiments, both **Move(B,C)** and **Move(A,B)** were treated as phantom resources. This led to redundancy in the resource pricing and iterations tended to cycle rather than converging. The choice of appropriate phantom resources to represent constraints appears to be critical.

Resource pricing is most effective for large problems where there are many instances of resources. There is little reason to expect it to do well on small blocks world problems that are dominated by goal interactions. Nevertheless, resource pricing does surprisingly well on these problems—giving considerable hope that it will be effective for large organizational planning problems where there are many instances of most resources and relatively few goal interactions that require the pricing of single instances of phantom resources.

7. A Structure for Utility Functions

In the blocks world problems of the last section, there was a utility associated with achieving each goal and there were a set of constraints that had to be satisfied. In organizational planning problems, there are many soft constraints. Soft constraints reduce the acceptability of a plan that violates the constraint but do not make the plan invalid. Essentially, soft constraints involve a penalty against utility for violating the constraint, but it may be preferable to violate the constraint rather than lose the utility that comes with accomplishing an additional goal.

Most organizational planning constraints are actually soft and there is a very fine line between hard and soft constraints. Users will often describe hard constraints on the problem solution, but when asked whether they would ever violate the constraint, they will find situations in which the constraint can be violated. It is really a soft constraint with a strong penalty for violating it. Problems should be represented with a smooth transition between hard constraints that can't be violated and soft constraints that carry a large penalty when violated.

This section summarizes and updates work to define a structure for utility functions that allows the soft constraints of organizational planning problems to be represented effectively. A utility function that is practical for planning and scheduling problems should be a compromise between several conflicting goals:

- It should represent user preferences in a reasonably natural and direct way.
- It should be useful during heuristic search when evaluating partial solutions.
- It should enable effective variable and value ordering decisions during heuristic search.

7.1 Constraint Types and their Interaction with Utility

In approaching the utility function, it is useful to separate:

1. The utility of achieving a goal (performing a task).
2. Binary and low order constraints on the solution–both hard and soft constraints.
3. Resource capacity constraints and other high order constraints on the solution.

Resource constraints are common in practical applications and are not effectively handled by general purpose constraint satisfaction techniques. Resource constraints are complex, N-way constraints between many tasks. N is typically large.

Viewed from the perspective of search strategies, resource constraints are nasty in that they tend to prune a branch of the search tree only after it is almost fully expanded. For example, if there are k instances of a resource, simple propagation of this resource constraint does not have an impact until after the k^{th} assignment of that resource. Then it immediately has the dramatic effect of eliminating this resource from the options available to all the remaining tasks.

While resource constraints are not handled effectively by generic constraint satisfaction techniques, specialized heuristics are effective with resource constraints. A heuristic approach, which is now widely practiced in AI scheduling applications, projects resource contention using statistical look-ahead techniques and uses these contention estimates in variable and value ordering heuristics. Statistical look-ahead techniques have been used in work on Opis [Muscettola & Smith 87], Cortes [Fox et al. 90, Sycara et al. 90] and Micro-Boss [Sadeh 91] and in work on Rome Laboratory's Advanced Planning System (APS) [APS 89].

7.2 Utility Functions in the Form $\Sigma u(t)*mod(t,...)$

These and other considerations lead to utility functions that are structured in the form $\Sigma u(t)*mod(t,...)$ where u(t) is a function of how and when a single task is completed and mod(t,...) captures the effects of dependencies between tasks. The mod(t,...) factor is a function of all the assignments made to t and to all tasks that are involved in binary or other low order constraints with t. The value of mod(t,...) should be 0 when the assignments made to t and related tasks violate a hard constraint. A soft constraint is represented by a value between 0 and 1. Values outside the range [0,1] can also be meaningful. This approach supports continuity between hard and soft constraints while still recognizing a difference between hard and soft constraints. For more details, see [Linden 91].

Typically, mod(t,...) is structured as a product where each factor in the product captures the effect of one constraint between t and other tasks. A product is one specific way of combining the effect of multiple soft constraint violations. Other combining functions are possible. For more details on this approach, see [Linden 91].

A utility function in the proposed form is a fairly natural way of representing real problems. The u(t) factor combines the effects of multiple evaluation criteria that depend only on the assignments made to the parameters of this single task (e.g., resource costs, timeliness of task completion, appropriateness of the resources for the task, etc.). The effect of dependencies between tasks (for example, one task must be performed before another) are captured in the mod(t,...) factor. Essentially, each hard or soft constraint between tasks becomes a factor in the mod(t,...) component of each of the constrained tasks. The effects of conjunctive and disjunctive goals can also be captured in the mod(t,...) component to handle the cases where goals have a conjunctive all-or-none

property or where goals are alternatives with decreasing additional value once M out of N are accomplished.

7.3 Evaluating Partial Solutions

The proposed structure for utility functions allows many equivalent formulations, and often one formulation is more useful than others when evaluating partial solutions. For example, consider the case of two tasks t1 and t2 where t1 establishes a precondition for t2. Assume that t2 achieves 10 units on the utility scale, and t1 has no independent utility except for its role in enabling t2. The utility function for these two tasks would then be 0*mod(t1,t2) + 10*mod(t2,t1) where mod(t1,t2) and mod(t2,t1) are 1 if t1 is accomplished before t2 (and maintained) and 0 otherwise. When evaluating a partial solution, this utility function gives no importance to t1; however, an equivalent utility function is x*mod(t1,t2) + (10-x)*mod(t2,t1). By choosing appropriate values for x, this form of the utility function can decompose the problem of choosing parameter values for t1 and t2 and support more effective least commitment strategies.

8. Heuristic Search with Non-Additive Utility

For resource pricing to be effective in practice, prices must converge rapidly toward an equilibrium. To the extent that individual planning cells have a significant effect on prices, which is often the case when phantom resources represent dependencies in the plan, it is useful to look for better ways to communicate about prices and get them to converge. We explored the idea that planning cells should bid for resources by giving a probability distribution about the price they will be willing to pay or receive. When there is uncertainty about what other resources will cost and about what the best local plan will be, the local planning cell has only probabilistic information about the price it will want to offer for this resource. A large amount of previous work on statistical look-ahead techniques for AI scheduling has explored bidding for resources in terms of subjective probabilities of use [Muscettola & Smith 87, APS 89, Sadeh & Fox 89, Sycara et al. 90, Sadeh 91, Sadeh and Fox 91, Johnston 92]. We propose bidding in terms of both the price to be paid and the probability of use at that price. This approach increases the communication bandwidth between distributed planning cells, is more likely to converge when there are conjunctive and disjunctive subgoals, and may lead to useful results even when convergence cannot be guaranteed. This section summarizes the current state of research on a theoretical framework for this approach using concepts from decision theoretic planning.

This probabilistic approach to pricing will be used both to choose ways to extend a partial plan (value ordering decisions) and choose which part of the plan to extend next (variable ordering decisions.

8.1 Variable and Value Ordering Search Heuristics

Most variable and value ordering heuristics used in heuristic search are sensitive to only one feature of the search state; for example, the minimum domain variable ordering heuristic finds the variable with the smallest domain of feasible values. But complex resource-bounded planning problems involve optimization in the presence of constraints. A fixed heuristic that is sensitive only to constraints and ignores utility considerations involves a discontinuity between hard and soft constraints and will be effective only for problems where hard constraints are the critical feature. Heuristics should depend on the utility function as well as the constraints.

What is needed is a general way to merge all the evidence available when making variable and value ordering decisions during search. This evidence comes from the utility function, binary constraints, and resource constraints. A decision theoretic viewpoint seems to provide justifiable semantics for computations that combine this evidence. A utility function in the form suggested above with separate components for the independent utility of the task and for the effects of interactions between tasks supports this decision theoretic approach.

The variable and value ordering decisions that need to be made repeatedly during search can be viewed as problems in decision theory where evidence for each decision comes from the problem's features and the current state of the problem solving. While heuristics are frequently thought of as inexpensive computations, any computation that is not exponential in problem size can be a useful heuristic, and simpler heuristics can always be selected once a generic approach is understood.

8.2 Past Research on Probabilistic Computations of Heuristics

Many domain-independent heuristics have been proposed to solve constraint satisfaction problems (CSP). These include the variable with smallest feasible domain, the most constraining variable, the least constraining value, and the value that participates in the most solutions to a relaxation of the problem. Work by Hansson et al. [92] computes a combination of heuristics that is effective for a specific CSP. Heuristics for constrained optimization problems should extend these CSP heuristics.

When there is a utility function as well as constraints, a key question is how to combine the evidence from the utility function and the constraints; for example, how much extra utility is needed to compensate for consuming a highly constrained resource. Sycara et al. [90] addressed this question by experimenting first with the two extreme cases (picking the value that gives the most utility for this task vs. picking the value that most reduces resource contention). A compromise is probably better than either extreme. Rather than trying to find the most appropriate compromise through experiments, theoretical results may be able to predict the compromise that is most effective overall.

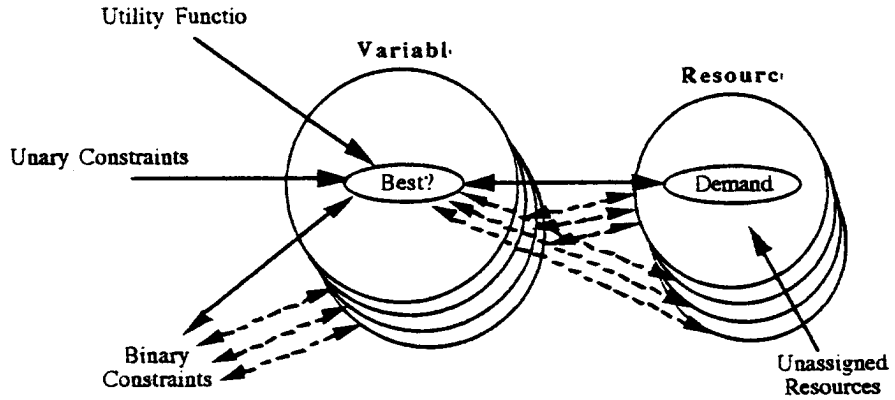


Figure 8-1: Sources of Evidence for Variable and Value Ordering Decisions.

Figure 8-1 is a simplified representation of the recent state of research on using probabilities to combine evidence when making variable and value ordering decisions during search. At any intermediate stage of the search process, one considers the remaining tasks (variables) and the remaining resources (values)—each shown as stacks of circles in the figure. Evidence for the variable and value

ordering decisions comes from the utility function, the unary constraints, binary constraints, and from statistics about resource contention. The heavy arrows show the flows of evidence involving the first variable and the first resource; the dashed arrows indicate other flows of evidence to and from the other variables and resources.

In Figure 8-1, for each variable there is a probability distribution (called **Best?**) that captures the available evidence about which resource will turn out to be the best choice for assigning to this variable. For each resource there is a probability distribution (called **Demand**) that summarizes the demand for that resource from all the variables. The double arrow between **Best?** and **Demand** characterizes one of the problems that needs to be solved: the **Best?** distribution is used to project statistics about resource demand, and the **Demand** for a resource influences the probability that a resource is the best global choice for assignment to a variable. Similarly, binary constraints involve evidence flowing in both directions between two variables—probabilities about assignments to one variable influence the probabilities about the best assignments to the other variable.

8.3 Overview of Approach

Figure 8-2 elaborates Figure 8-1 with two additional concepts: the **Net Utility** of each potential assignment of a value to a variable and the **Marginal Utility** of a resource. The double arrow between **Best?** and **Demand** is eliminated, but there are still cycles in the influences between the probability distributions shown in Figure 8-2. The elimination of these cycles is discussed in Section 8.8 and depicted in Figure 83.

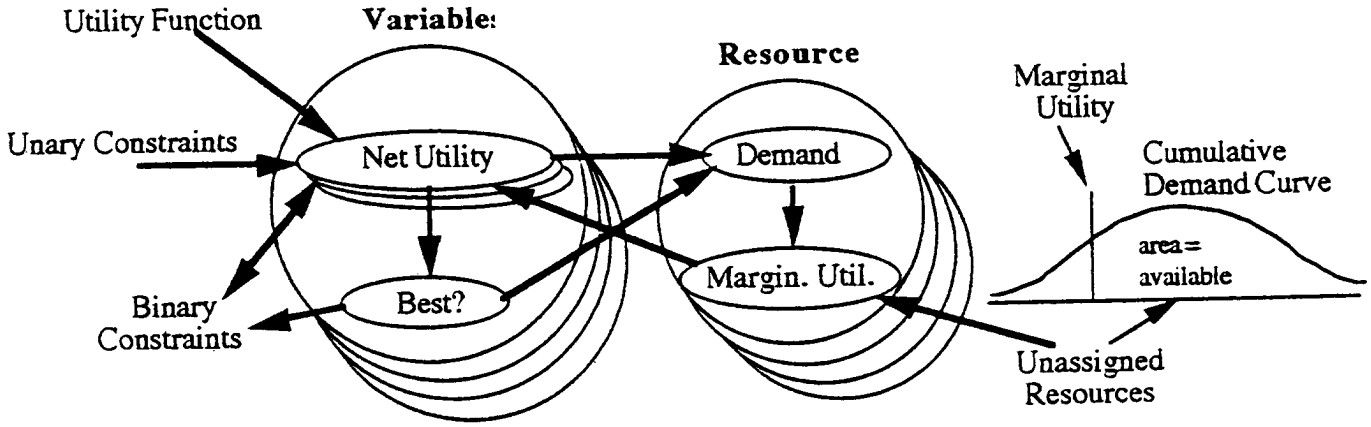


Figure 8-2: Evidence for Each Option is Accumulated in Probability Distributions about Net Utility.

8.4 Net Incremental Utility

The first key concept is **net incremental utility**—or more simply the **net utility**. Intuitively, the net utility of a potential assignment of a value to a variable is the

utility of that assignment less the opportunity loss for other variables caused by consumption of the resource and by constraints involving the assigned variable.

The net utility cannot be known exactly without a full search, but it can be estimated. Uncertainty about the net utility is subjective. Net utility can be calculated in exponential time, but with a computation that uses less than exponential time, its value is uncertain. With each variable and each possible value assignment to the variable, one associates a probability distribution that captures the available evidence about the net utility that will result if this value is chosen for this variable. The net utility that is being estimated is defined as follows:

- The **net incremental utility** of assigning a value to a variable, when given an existing partial problem solution, is the difference between the best complete extension that includes that assignment and the best complete extension that assigns no resource to that variable.

- More precisely, **net incremental utility** is relative to the search strategy that is being used. When the search strategy being used is not guaranteed to find the best solution, then net incremental utility is defined relative to the expected values of the best solution that will be found by the given search strategy. Thus, given a search strategy, the net incremental utility of assigning a resource to an operation is the difference between the expected value of the best complete solution that will be found by following that search strategy after including that assignment less the expected value of the best complete solution that will be found by following that search strategy after assigning no resource to that operation.

8.5 Net Utility and the "Best?" Choice

The unary constraints and the u(t) component of the utility function provide a local estimate of utility. This local estimate is the initial evidence for the probable net incremental utility. When a utility function is additive, the local utility comes directly from the utility function. Since a probability distribution about the local utility is all that is needed, the additivity assumption is not required. Most utility functions—as long as they have some partially additive structure—yield a probability distribution about the utility that will be achieved by assigning a value to a variable.

Binary constraints and resource constraints also influence estimates of the net incremental utility. Before dealing with these constraints, we need to derive the probability that an assignment is the best choice from distributions about net incremental utility.

By reformulating the problem to focus on net incremental utility, the probability that an assignment to a variable is the best choice becomes a derived concept rather than an intuitive concept as in [Sadeh & Fox 89, 90, Sadeh 91].

Best? is a probability distribution over the possible resource assignments that is computed as the probability that a random utility value selected from the net utility distribution about the resource assignment is bigger than any other such utility value.

The variances in the probability distributions about net utility control whether one assignment is slightly or dramatically better than alternatives.

8.6 Computing Marginal Utility

The next issue is the interaction between assignments to variables and the projection of resource contention statistics. Previous work has used the probability that an assignment is the best choice to project demand for resources, but demand also influences the probabilities about the best choice. A key insight derives from Wellman's work on applying economic theory to transportation scheduling [Wellman 92]. The marginal utility of a resource establishes a price for the resource, and an agent evaluating the options for assignment to a variable should favor using a resource to the extent that the local utility of using a resource exceeds the globally determined price of the resource. Choosing the best resource to assign to a variable is no longer based on the relative size of the local utilities; rather it is based on the amount by which the local utility exceeds its resource's price.

8.7 Propagating the Influence of Binary Constraints

Binary constraints between variables also influence the net utility computations. Processing evidence from binary constraints can be thought of as an extension of arc consistency concepts from CSP problems. The utility function makes the constraint propagation more complex. The effect of constraints on probabilities about the best choice has been studied by Muscettola & Smith [87] and Sadeh & Fox [89, 90]. Net utility should include the utility lost when an assignment constrains the utility achievable by a dependent variable. For each assignment that a related variable may make, a variable estimates the utility it will lose and communicates that estimated loss to the dependent variable.

In addition to estimating the utility lost by other variables, it may also be useful to estimate the degree to which an assignment restricts the choices available for other variables. Consideration is being given to using entropy concepts to measure the distance between a partial solution and a complete solution or between two consistent partial solutions. Computing the entropy of a partial solution is often straightforward. The entropy involved in each Best? distribution may capture the flexibility that is left to make assignments to that variable. When a value assignment to one variable decreases the entropy of another variable that it constrains, this may be taken as a measure of the constraining effect of that proposed value assignment. These may be useful measures that generalize the most constraining variable and least constraining value heuristics.

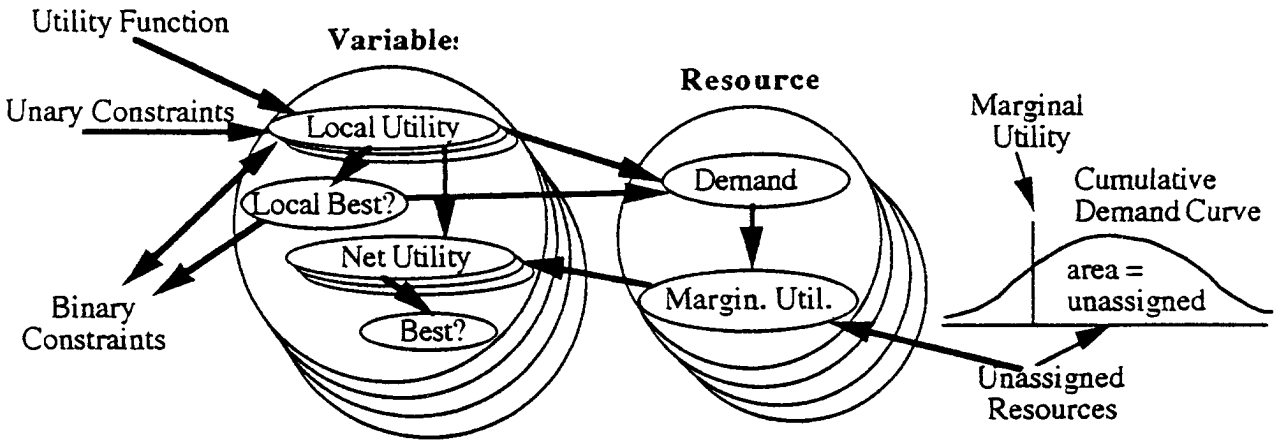


Figure 8-3: Additional distributions are introduced to break cycles and avoid rumor propagation.

8.8 Rumor Control and Convergence

Figure 8-2 showed a cycle in the evidence being passed from net utility to resource demand to marginal utility and back to net utility. This cycle needs to be broken by using the standard approach to rumor control from Bayesian nets. Essentially, the net utility information that the variable's agent passes to the resource agent must not include previous evidence received from the resource agent. The same restriction applies to evidence passed to other variable agents through the binary constraints. Figure 8-3 is a more detailed version of Figure 8-2 showing that the local net utility and local best choice (which do not reflect evidence received from the resource agents) is computed and passed to the resource agents. While not shown in Figure 8-3, the information passed between variable agents through the binary constraints must also be restricted so evidence previously received from another agent is excluded from all evidence passed to that agent.

There are still some longer cycles in the information flows. Intuition says they are not significant; however, further research is needed to derive conditions under which they can be proven to be insignificant.

8.9 Experimental results

Initial experiments to test the decision-theoretic computations for variable and value ordering decisions are reported in [Linden & Vrotney 92]. The experiments use a statistical look-ahead algorithm that starts from a utility function, projects probable demand for each resource, computes a probable marginal utility for each resource, and uses these probabilities to make variable and value ordering choices during search. Initial experiments tested these concepts on assignment problems and compared the solution found on the first branch explored using these heuristics with the solution found by a simple

greedy algorithm. The statistical look-ahead algorithm found solutions that average 4-7% higher utility. When an optimal solution was known, the statistical look-ahead, almost always either found it on the first branch or was within 0.5% of optimal The frequency of better results improved on larger problems. The statistical look-ahead improved the result (relative to greedy on 85% of the small problems (16 tasks and 12 resources), on 95% of the medium size problems (30 tasks and 24 resources) and on all runs of the larger problems (56 tasks and 48 resources).

9. Conclusions

This report argues that resource pricing is a promising way to coordinate planning by multiple planning cells. The techniques proposed are especially relevant for large organizational planning problems like military crisis action planning.

Resource pricing is an established way to allocate resources efficiently, and much of what is needed is solve organizational planning problems is resource allocation. However, this resource allocation occurs in the context of discrete goals, constraints, and interactions of the kind addressed by traditional AI planning technology. This research developed techniques to combine resource pricing mechanisms with traditional AI planning technology.

The combination of techniques is especially useful when there are many goals that may be contingent, uncertain, changing, and imperfectly perceived. Trade-offs between goals are formalized in utility functions that are not required to be additive but do have some additive structure. Concepts from decision theory provide a foundation for the statistical look-ahead techniques used to make variable and value ordering decisions during heuristic search. The theory will extend current practice to handle resource-bounded planning problems with complex, non-additive utility functions and both hard and soft constraints.

E. References

[Alterman 86] R. Alterman. An Adaptive Planner. *Proc. Fifth National Conference on Artificial Intelligence.* Morgan Kaufman Publ., 1986, pp. 65-71.

[Alterman 88] R. Alterman. Adaptive Planning,. Cognitive Science. 12, 1988 pp.393-421.

[APS 89] Software Design Document for the Advanced Planning System. Unisys Corporation, Prepared for Rome Air Development Center, Griffiss Air Force Base, NY. CDRL B012, 1989.

[Bellman 57] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, N.J., 1957

[Brooks 87] R. Brooks. Planning is Just a Way of Avoiding Figuring Out What to Do Next." Tech Rpt. Working Paper 303, MIT AI Laboratory, MIT 1987.

[Einav 91] D. Einav. Reasoning Automation under Resource Constraints. Ph.D. Thesis, Laboratory for Intelligent Systems, Dept Engineering-Economic Systems, Stanford Univ., December, 1991.

[Fox et al. 89] Mark S. Fox, Norman Sadeh, & Can Baykan, "Constrained Heuristic Search," AAAI-89, pp. 309-315.

[Hansson et al 92] Othar Hansson, Gerhard Holt, and Andrew Mayer, "Experiments with a Decision-Theoretic Scheduler," 1992 AAAI Spring Symposium on Practical Approaches to Scheduling and Planning, NASA Ames TR FIA-92-17, May 1992.

[Johnston 92] Mark D. Johnston, "Spike: AI Scheduling for Hubble Space Telescope after 18 Months of Orbital Operations." AAAI Spring Symposium: Practical Approaches to Scheduling and Planning, Working Notes, March 1992.

[Linden & Glicksman 87] Theodore A. Linden and Jay Glicksman , "Contingency Planning for an Autonomous Land Vehicle," Proc. IJCAI-87, Morgan Kaufman Publ., Vol. 10.

[Linden 89] Theodore A. Linden, "Planning by Transformational Synthesis," *IEEE Expert*, 4,2, Summer, 1989.

[Linden 90] Theodore A. Linden, "Transformational Synthesis: An Approach to Large-Scale Planning Applications," DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control., Morgan Kaufman Publ., San Mateo, CA, Nov. 1990

[Linden 91] Theodore A. Linden, "Preference-directed, Co-operative Resource Allocation and Scheduling." Final Technical Report, DARPA Order No. 6685, Advanced Decision Systems Report TR-1270-3, Sept. 1991.

[Linden & Vrotney 92] Theodore A. Linden and William Vrotney, "Transformational Planning for Resource Constrained Problems," Annual Technical Report to DARPA, Advanced Decision Systems Report TR-18246-1, November. 1992.

[Muscettola & Smith 87] Nicola Muscettola and Stephen F. Smith, "A Probabilistic Framework for Resource-Constrained Multi-agent Planning." Proc. Tenth Inter. Joint Conf. on Artificial Intelligence, Morgan Kaufman, Publ. 1987, pp. 1063-1066.

[Sadeh & Fox 89] Norman Sadeh and Mark S. Fox, Preference Propagation in Temporal/Capacity Constraint Graphs. The Robotics Institute, Carnegie Mellon Univ., CMU-RI-TR-89-2, 1989.

[Sadeh & Fox 90] Norman Sadeh and Mark S. Fox, "Variable and Value Ordering Heuristics for Activity-based Job-shop Scheduling." Proc. of the Fourth Inter. Conf. on Expert Systems in Production and Operations Management, 1990, pp 134-144,.

[Sadeh 91] Norman Sadeh, "Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling." PhD Thesis, School of Computer Science, Carnegie Mellon University, 1991.

[Sadeh and Fox 91] Norman Sadeh & Mark S. Fox, "Variable and Value Ordering Heuristics for Hard Constraint Satisfaction Problems: An Application to Job Shop Schedulilng." Technical Report CMU-RI-TR-91-23, The Robotics Institute, Carnegie Mellon University, Nov. 1991.

[Schoppers 87] M. Schoppers, "Universal Plans for Reatcive Robots in Unpredictable Environments," Proc. IJCAI-87, Morgan Kaufman Publ., Vol. 10, 1987, pp. 852-859.

[Sycara et al. 90] Katia P. Sycara, S. Roth, N. Sadeh, and M. Fox, "Managing Resource Allocation in Multi-Agent Time-constrained Domains." DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control., Morgan Kaufman Publ., San Mateo, CA, Nov. 1990, pp. 240-250.

[Wellman 92] Michael P. Wellman, "A General-Equilibrium Approach to Distributed Transporation Planning." AAAI-92, AAAI Press, 1992.

DISTRIBUTION LIST

addresses	number of copies
JOHN F. LEMMER ROME LABORATORY/C3CA 525 BROOKS ROAD GRIFFISS AFB NY 13441-4505	5
KESTREL INSTITUTE 3260 HILLVIEW AVENUE PALO ALTO CA 94304	5
RL/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY GRIFFISS AFB NY 13441-4514	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER DTIC-FDAC CAMERON STATION BUILDING 5 ALEXANDRIA VA 22304-6145	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
NAVAL WARFARE ASSESSMENT CENTER GIDEP OPERATIONS CENTER/CODE QA-50 ATTN: E RICHARDS CORONA CA 91718-5000	1
WRIGHT LABORATORY/AAAI-2 ATTN: MR FRANKLIN HUTSON WRIGHT-PATTERSON AFB OH 45433-6543	1
AFIT/LDEE 2950 P STREET WRIGHT-PATTERSON AFB OH 45433-6577	1

WRIGHT LABORATORY/MTEL 1
WRIGHT-PATTERSON AFB OH 45433

AAMRL/HE 1
WRIGHT-PATTERSON AFB OH 45433-6573

AUL/LSE 1
BLDG 1405
MAXWELL AFB AL 36112-5564

US ARMY STRATEGIC DEF 1
CSSD-IM-PA
PO BOX 1500
HUNTSVILLE AL 35807-3801

COMMANDING OFFICER 1
NAVAL AVIONICS CENTER
LIBRARY D/765
INDIANAPOLIS IN 46219-2189

COMMANDING OFFICER 1
NCCOSC RDTE DIVISION
CODE 02748, TECH LIBRARY
53560 HULL STREET
SAN DIEGO CA 92152-5001

CMDR 1
NAVAL WEAPONS CENTER
TECHNICAL LIBRARY/C3431
CHINA LAKE CA 93555-6001

SPACE & NAVAL WARFARE SYSTEMS COMM 1
WASHINGTON DC 20363-5100

CDR, U.S. ARMY MISSILE COMMAND 2
REDSTONE SCIENTIFIC INFO CENTER
AMSMI-RD-CS-R/ILL DOCUMENTS
REDSTONE ARSENAL AL 35898-5241

ADVISORY GROUP ON ELECTRON DEVICES 2
ATTN: DOCUMENTS
2011 CRYSTAL DRIVE,SUITE 307
ARLINGTON VA 22202

REPORT COLLECTION, RESEARCH LIBRARY 1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

AEDC LIBRARY 1
TECH FILES/MS-100
ARNOLD AFB TN 37389

COMMANDER/USAISC 1
ATTN: ASOP-DO-TL
BLDG 61801
FT HUACHUCA AZ 85613-5000

AIR WEATHER SERVICE TECHNICAL LIB 1
FL 4414
SCOTT AFB IL 62225-5458

AFIWC/MSO 1
102 HALL BLVD STE 315
SAN ANTONIO TX 78243-7016

SOFTWARE ENGINEERING INST (SEI) 1
TECHNICAL LIBRARY
5000 FORBES AVE
PITTSBURGH PA 15213

DIRECTOR NSA/CSS 1
W157
9800 SAVAGE ROAD
FORT MEADE MD 21055-6000

NSA 1
ATTN: D. ALLEY
DIV X911
9800 SAVAGE ROAD
FT MEADE MD 20755-6000

DOD 1
R31
9800 SAVAGE ROAD
FT. MEADE MD 20755-6000

DIRNSA 1
R509
9800 SAVAGE ROAD
FT MEADE MD 20775

ESC/IC 1
50 GRIFFISS STREET
HANSCOM AFB MA 01731-1619

FL 2807/RESEARCH LIBRARY 1
OL AA/SULL
HANSCOM AFB MA 01731-5000

TECHNICAL REPORTS CENTER 1
MAIL DROP D130
BURLINGTON ROAD
BEDFORD MA 01731

DEFENSE TECHNOLOGY SEC ADMIN (DTSA) 1
ATTN: STTD/PATRICK SULLIVAN
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

DARPA/TTO 1
ATTN: DV
1400 WILSON BLVD
ARLINGTON VA 22203-2309

MS. KAREN ALGUIRE 1
RL/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

JAMES ALLEN 1
COMPUTER SCIENCE DEPT/BLDG RM 732
UNIV OF ROCHESTER
WILSON BLVD
ROCHESTER NY 14627

YIGAL ARENS
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

MR. RAY BAREISS
THE INST. FOR LEARNING SCIENCES
NORTHWESTERN UNIV
1890 MAPLE AVE
EVANSTON IL 60201 1

MR. JEFF BERLINER
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 1

MARIE A. BIENKOWSKI
SRI INTERNATIONAL
333 RAVENSWOOD AVE/EK 337
MENLO PRK CA 94025 1

DR MARK S. BODDY
HONEYWELL SYSTEMS & RSCH CENTER
3660 TECHNOLOGY DRIVE
MINNEAPOLIS MN 55418 1

PIERO P. BONISSONE
GE CORPORATE RESEARCH & DEVELOPMENT
BLDG K1-RM 5C-32A
P. O. BOX 8
SCHENECTADY NY 12301 1

MR. DAVID BROWN
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269 1

MR. MARK BURSTEIN
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 1

MR. GREGG COLLINS
INST FOR LEARNING SCIENCES
1890 MAPLE AVE
EVANSTON IL 60201 1

MR. RANDALL J. CALISTRI-YEH 1
ORA CORPORATION
301 DATES DRIVE
ITHACA NY 14850-1313

DR STEPHEN E. CROSS 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

MS. JUDITH DALY 1
ARPA/ASTO
3701 N. FAIRFAX DR., 7TH FLOOR
ARLINGTON VA 22203-1714

THOMAS CHEATHAM 1
HARVARD UNIVERSITY
DIV OF APPLIED SCIENCE
AIKEN, RM 104
CAMBRIDGE MA 02138

MS. LAURA DAVIS 1
CODE 5510
NAVY CTR FOR APPLIED RES IN AI
NAVAL RESEARCH LABORATORY
WASH DC 20375-5337

MS. GLADYS CHOW 1
COMPUTER SCIENCE DEPT.
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

THOMAS L. DEAN 1
BROWN UNIVERSITY
DEPT OF COMPUTER SCIENCE
P.O. BOX 1910
PROVIDENCE RI 02912

WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

MR. ROBERTO DESIMONE 1
SRI INTERNATIONAL (EK335)
333 RAVENSWOOD AVE
MENLO PRK CA 94025

PAUL R. COHEN
UNIV OF MASSACHUSETTS
COINS DEPT
LEDERLE GRC
AMHERST MA 01003 1

MS. MARIE DEJARDINS
SRI INTERNATIONAL
333 RAVENSWOOD AVENUE
MENLO PRK CA 94025 1

JON DOYLE
LABORATORY FOR COMPUTER SCIENCE
MASS INSTITUTE OF TECHNOLOGY
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139 1

DR. BRIAN DRABBLE
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH/80 S. BRIDGE
EDINBURGH EH1 LHN
UNITED KINGDOM 1

MR. SCOTT FOUSE
ISX CORPORATION
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361 1

MR. STU DRAPER
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269 1

MARK FOX
DEPT O INDUSTRIAL ENGRG
UNIV OF TORONTO
4 TADDLE CREAK ROAD
TORONTO, ONTARIO, CANADA 1

MR. GARY EDWARDS
4353 PARK TERRACE DRIVE
WESTLAKE VILLACA 91361 1

MS. MARTHA FARINACCI
MITRE
7525 COLSHIRE DRIVE
MCLEAN VA 22101 1

MR. RUSS FREW 1
GENERAL ELECTRIC
MOORESTOWN CORPORATE CENTER
BLDG ATK 145-2
MOORESTOWN NJ 08057

MICHAEL FEHLING 1
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305

MR. RICH FRITZSON 1
CENTER OR ADVANCED INFO TECHNOLOGY
UNISYS
P.O. BOX 517
PAOLI PA 19301

MR KRISTIAN J. HAMMOND 1
UNIV OF CHICAGO
COMPUTER SCIENCE DEPT/RY155
1100 E. 58TH STREET
CHICAGO IL 60637

MR. ROBERT FROST 1
MITRE CORP
WASHINGTON C3 CENTER, MS 644
7525 COLSHIER ROAD
MCLEAN VA 22101-3481

RICK HAYES-ROTH 1
CIMFLEX-TEKNOWLEDGE
1810 EMBARCADERO RD
PALO ALTO CA 94303

RANDY GARRETT 1
INST FOR DEFENSE ANALYSES (IDA)
1801 N. BEAUREGARD STREET
ALEXANDRA VA 22311-1772

MR. JIM HENDLER 1
UNIV OF MARYLAND
DEPT OF COMPUTER SCIENCE
COLLEGE PARK MD 20742

MS. YOLANDA GIL 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

MR.MAX HERION
ROCKWELL INTERNATIONAL SCIENCE CTR
444 HIGH STREET
PALO ALTO CA 94301 1

MR. STEVE GOYA
DISA/JIEO/GS11
CODE TBD
11440 ISAAC NEWTON SQ
RESTON VA 22090 1

MR. MORTON A. HIRSCHBERG, DIRECTOR
US ARMY RESEARCH LABORATORY
ATTN; AMSRL-CI-CB
ABERDEEN PROVING GROUND MD
21005-5066 1

MR. MARK A. HOFFMAN
ISX CORPORATION
1165 NORTHCHASE PARKWAY
MARIETTA GA 30067 1

MR. RON LARSEN
NAVAL CMD, CONTROL & OCEAN SUR CTR
RESEARCH, DEVELOP, TEST & EVAL DIV
CODE 444
SAN DIEGO CA 92152-5000 1

DR. JAMES JUST
MITRE
DEPT. W032-M/S Z360
7525 COLSHIER RD
MCLEAN VA 22101 1

MR. CRAIG KNOBLOCK
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

MR. RICHARD LOWE (AP-10)
SRA CORPORATION
2000 15TH STREET NORTH
ARLINGTON VA 22201 1

MR. TED C. KRAL
BBN SYSTEMS & TECHNOLOGIES
4015 HANCOCK STREET, SUITEE 101
SAN DIEGO CA 92110 1

MR. JOHN LOWRENCE 1
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025

DR. ALAN MEYROWITZ 1
NAVAL RESEARCH LABORATORY/CODE 5510
4555 OVERLOOK AVE
WASH DC 20375

ALICE MULVEHILL 1
MITRE CORPORATION
BURLINGTON RD
M/S K-302
BEDFORD MA 01730

ROBERT MACGREGOR 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL REY CA 90292

WILLIAM S. MARK, MGR AI CENTER 1
LOCKHEED MISSILES & SPACE CENTER
1801 PAGE MILL RD
PALO ALTO CA 94304-1211

RICHARD MARTIN 1
SOTWARE ENGINEERING INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 16213

DREW MCDERMOTT 1
YALE COMPUTER SCIENCE DEPT
P.O. BOX 2158, YALE STATION
51 PROPSPECT STREET
MEW HAVEN CT 06520

MS. CECILE PARIS 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

DOUGLAS SMITH 1
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304

DR. AUSTIN TATE
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH
80 SOUTH BRIDGE
EDINBURGH EH1 1HN - SCOTLAND

1

EDWARD THOMPSON
ARPA/SISTO
3701 N. FAIRFAX DR., 7TH FL
ARLINGTON VA 22209-1714

1

MR. STEPHEN F. SMITH
ROBOTICS INSTITUTE/CMU
SCHENLEY PRK
PITTSBURGH PA 15213

1

DR. ABRAHAM WAKSMAN
AFOSR/NM
110 DUNCAN AVE., SUITE B115
BOLLING AFB DC 20331-0001

1

JONATHAN P.STILLMAN
GENERAL ELECTRIC CRD
1 RIVER RD, RM K1-5C31A
P. O. BOX 8
SCHENECTADY NY 12345

1

MR. EDWARD C. T. WALKER
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

1

MR. BILL SWARTOUT
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

1

GIO WIEDERHOLD
STANFORD UNIVERSITY
DEPT OF COMPUTER SCIENCE
438 MARGARET JACKS HALL
STANFORD CA 94305-2140

1

KATIA SYCARA/THE ROBOTICS INST
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIV
DOHERTY HALL RM 3325
PITTSBURGH PA 15213

MR. DAVID E. WILKINS
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PARK CA 94025 — 1

DR. PATRICK WINSTON
MASS INSTITUTE OF TECHNOLOGY
RM NE43-817
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139 — 1

HUA YANG
COMPUTER SCIENCE DEPT
UNIV OF CALIORNIA
LOS ANGELES CA 90024 — 1

LTCOL DAVE NEYLAND
ARPA/ISTO
3701 N. FAIRFAX DRIVE, 7TH FLOOR
ARLINGTON VA 22209-1714 — 1

MR. RICK SCHANTZ
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138 — 1

LTC FRED M. RAWCLIFFE
USTRANSCOM/TCJ5-SC
BLDG 1900
SCOTT AFB IL 62225-7001 — 1

ARPA/SISTO
ATTN: MR JOHN P. SCHILL
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714 — 1

MR. DONALD F. ROBERTS
RL/C3CA
525 BROOKS ROAD
GRIFFISS AFB NY 13441-4505 — 1

ALLEN SEARS
MITRE
7525 COLESHIRE DRIVE, STOP Z289
MCLEAN VA 22101 — 1

STEVE ROTH
CENTER FOR INTEGRATED MANUFACTURING
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 15213-3890 1

JEFF ROTHENBERG
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA CA 90407-2138 1

YOAV SHOHAM
STANFORD UNIVERSITY
COMPUTER SCIENCE DEPT
STANFORD CA 94305 1

MR. DAVID B. SKALAK
UNIV OF MASSACHUSETTS
DEPT OF COMPUTER SCIENCE
RM 243, LGRC
AMHERST MA 01003 1

MR. MIKE ROUSE
AFSC
7800 HAMPTON RD
NORFOLK VA 23511-6097 1

MR. DAVID E. SMITH
ROCKWELL INTERNATIONAL
444 HIGH STREET
PALO ALTO CA 94301 1

JEFF ROTHENBERG
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MIN STREET
SANTA MONICA CA 90407-2138 1

DR LARRY BIRNBAUM
NORTHWESTERN UNIVERSITY
ILS
1890 MAPLE AVE
EVANSTON IL 60201 1

MR RANDALL J. CALISTRI-YEH
ORA
301 DATES DR
ITHACA NY 14850-1313 1

MR WESLEY CHU
COMPUTER SCIENCE DEPT
UNIVERSITY OF CALIFORNIA
LOS ANGELES CA 9002 — 1

MR PAUL R COHEN
UNIVERSITY OF MASSACHUSETTS
COINS DEPT, LEDERLE GRC
AMHERST MA 01003 — 1

MR DON EDDINGTON
NAVAL COMMAND, CONTROL & OCEAN
SURV CENTER
RDT&E DIVISION, CODE 404
SAN DIEGO CA 92152-5000 — 1

MR. LEE ERMAN
CIMFLEX TECKNOWLEDGE
1810 EMBARCARDERO RD
PALO ALTO CA 94303 — 1

MR DICK ESTRADA
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138 — 1

MR HARRY FORSDICK
BBN SYSTEMS AND TECHNOLOGIES
10 MOULTON ST
CAMBRIDGE MA 02138 — 1

MR MATTHEW L. GINSBERG
CIRL, 1269
UNIVERSITY OF OREGON
EUGENE OR 97403 — 5

MR IRA GOLDSTEIN
OPEN SW FOUNDATION RESEARCH INST
ONE CAMBRIDGE CENTER
CAMBRIDGE MA 02142 — 1

MR MOISES GOLDSZMIDT
INFORMATION AND DECISION SCIENCES
ROCKWELL INTL SCIENCE CENTER
444 HIGH ST, SUITE 400
PALO ALTO CA 94301 — 1

MR JEFF GROSSMAN, CO
NCCOSC RDTE DIV 44
5370 SILVERGATE AVE, ROOM 1405
SAN DIEGO CA 92152-5146 1

JAN GUNTHER
ASCENT TECHNOLOGY, INC.
64 SIDNEY ST, SUITE 380
CAMBRIDGE MA 02139 1

DR LYNETTE HIRSCHMAN
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730 1

MS ADELE E. HOWE
COMPUTER SCIENCE DEPT
COLORADO STATE UNIVERSITY
FORT COLLINS CO 80523 1

DR LESLIE PACK KAELBLING
COMPUTER SCIENCE DEPT
BROWN UNIVERSITY
PROVIDENCE RI 02912 1

SUBBARAO KAMBHAMPATI
DEPT OF COMPUTER SCIENCE
ARIZONA STATE UNIVERSITY
TEMPE AZ 85287-5406 1

MR THOMAS E. KAZMIERCZAK
SRA CORPORATION
331 SALEM PLACE, SUITE 200
FAIRVIEW HEIGHTS IL 62208 1

PRADEEP K. KHOSLA
ARPA/SSTO
3701 N. FAIRFAX DR
ARLINGTON VA 22203 1

MR CRAIG KNOBLOCK
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292 1

DR CARLA LUDLOW 1
ROME LABORATORY/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

DR MARK T. MAYBURY 1
ASSOCIATE DIRECTOR OF AI CENTER
ADVANCED INFO SYSTEMS TECH G041
MITRE CORP, BURLINGTON RD, MS K-329
BEDFORD MA 01730

MR DONALD P. MCKAY 1
PARAMAX/UNISYS
P O BOX 517
PAOLI PA 19301

DR KAREN MYERS 1
AI CENTER
SRI INTERNTIONAL
333 RAVENSWOOD
MENLO PARK CA 94025

DR MARTHA E POLLACK 1
DEPT OF COMPUTER SCIENCE
UNIVERSITY OF PITTSBURGH
PITTSBURGH PA 15260

RAJ REDDY 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

EDWINA RISSLAND 1
DEPT OF COMPUTER & INFO SCIENCE
UNIVERSITY OF MASSACHUSETTS
AMHERST MA 01003

MR NORMAN SADEH 1
CIMDS
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIVERSITY
PITTSBURGH PA 15213

MR ERIC TIFFANY 1
ASCENT TECHNOLOGY INC.
237 LONGVIEW TERRACE
WILLIAMSTOWN MA 01267

MANUELA VELOSO 1
CARNEGIE MELLON UNIVERSITY
SCHOOL OF COMPUTER SCIENCE
PITTSBURGH PA 15213-3891

MR DAN WELD 1
DEPT OF COMPUTER SCIENCE & ENG
MAIL STOP FR-35
UNIVERSITY OF WASHINGTON
SEATTLE WA 98195

MR CRAIG WIER 1
ARPA/SISTO
3701 N. FAIRFAX DR
ARLINGTON VA 22203

MR JOE ROBERTS 1
ISX CORPORATION
4301 N FAIRFAX DRIVE, SUITE 301
ARLINGTON VA 22203

COL JOHN A. WARDEN III 1
ASC/CC
225 CHENNAULT CIRCLE
MAXWELL AFB AL 36112-6426

DR TOM GARVEY 1
ARPA/SISTO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

MR JOHN N. ENTZMINGER, JR. 1
ARPA/DIRO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

LT COL ANTHONY WAISANEN, PHD 1
COMMAND ANALYSIS GROUP
HQ AIR MOBILITY COMMAND
402 SCOTT DRIVE, UNIT 3L3
SCOTT AFB IL 62225-5307

DIRECTOR 1
ARPA/SISTO
3701 NORTH FAIRFAX DRIVE
ARLINGTON VA 22203-1714

MS LESLIE WILLIAMS
DIGITAL SYSEMS RSCH INC
4301 NORTH FAIRFAX DRIVE
SUITE 725
ARLINGTON VA 22203 — 1

DECISIONS & DESIGNS INC.
ATTN: ANN MARTIN
8219 LEESBURG PIKE
SUITE 390
VIENNA VA 22182 — 1

MS LEAH WONG
NCCOSC RDTE DIV
53570 SILVERGATE AVE
SAN DIEGO CA 92152-5246 — 5

OFFICE OF THE CHIEF OF NAVAL RSCH
ATTN: MR PAUL QUINN
CODE 311
800 N. QUINCY STREET
ARLINGTON VA 22217 — 1

NCCOSC RDTE DIV 404
ATTN: MR DON EDDINGTON
53560 HULL STREET
SAN DIEGO CA 92152-5001 — 1

BBN SYSTEMS AND TECHNOLOGY
ATTN: MR MAURICE MCNEIL
9655 GRANITE RIDGE DRIVE, SUITE 245
SAN DIEGO CA 92123 — 1

Rome Laboratory

Customer Satisfaction Survey

RL-TR-_____

Please complete this survey, and mail to RL/IMPS, 26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

_____

_____

Organization Name:_____(Optional)

Organization POC: _____(Optional)

Address:_____

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful 1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No_____

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.

*U.S. GOVERNMENT PRINTING OFFICE: 1995-710-126-20063

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.